

Credit: TROPOMI, ESA, Copernicus, KNMI



Herramientas de Python para Analizar Datos de NO₂

Pawan Gupta y Melanie Follette-Cook

Webinar Avanzado: Monitoreo de Alta Resolución del NO₂ desde el Espacio con TROPOMI
Mayo de 2019

Sesión 3

Introducción a Herramientas de Python para Datos del Tropospheric Monitoring Instrument* (TROPOMI)

- Leer un archivo NetCDF y aprender acerca de SDS**
- Leer y mapear datos de NO₂
- Leer y extraer datos de NO₂ en una ubicación particular
- Leer NetCDF y extraer datos a formato ASCII

*Instrumento de Monitoreo Troposférico en inglés

** SDS= Siglas de “Scientific Data Sets”, Conjuntos de datos científicos, en inglés

Objetivos de Aprendizaje

Para el final de esta presentación, usted podrá:

- Leer, extraer y mapear conjuntos de datos de NO₂ de TROPOMI

Conjuntos de Datos y Tareas

- **Datos**

- Datos OMI NO₂
- Datos TROPOMI NO₂

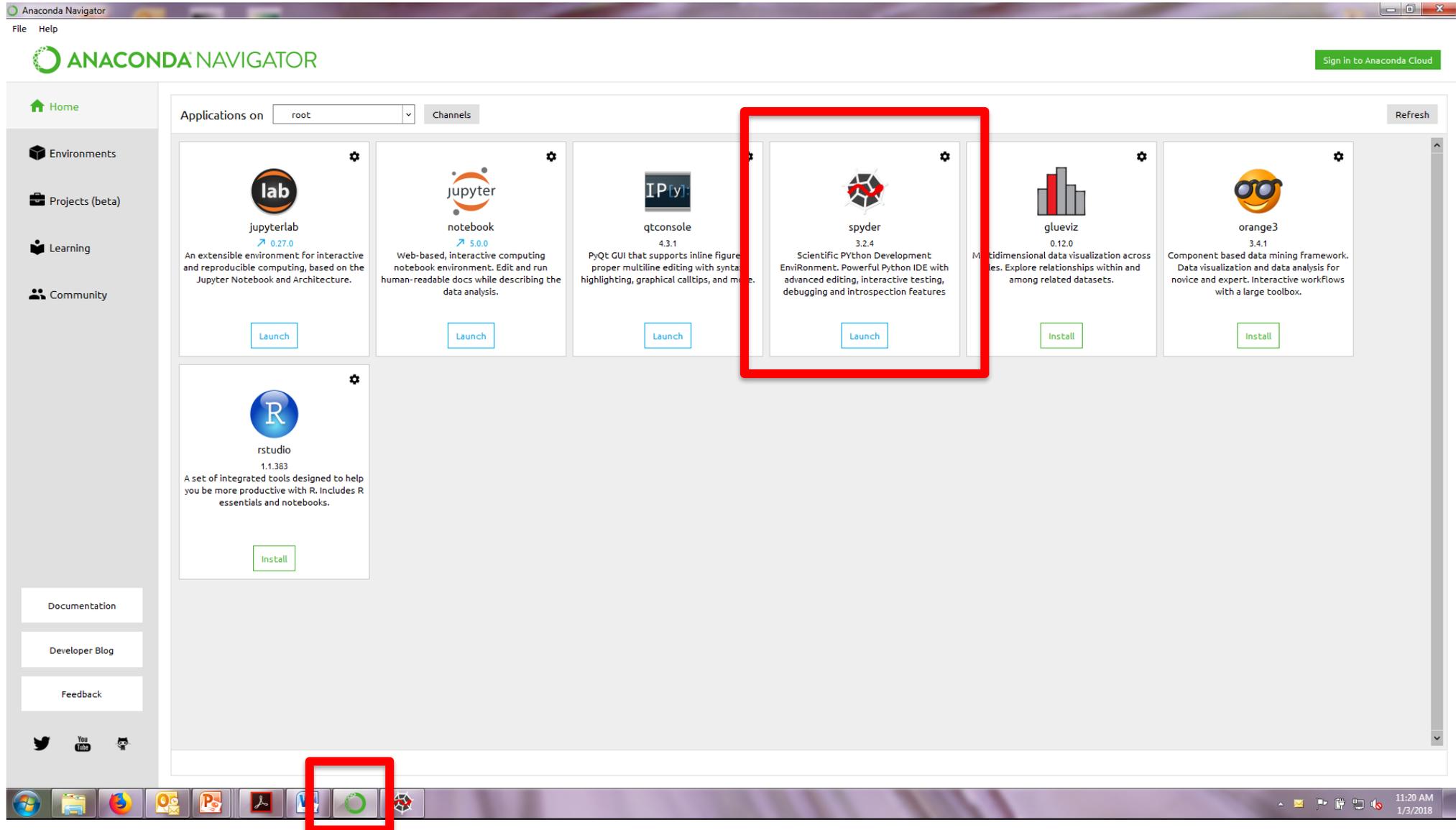
- **Tareas**

- Leer y listar conjuntos de datos científicos (scientific data sets o SDS)
- Leer y mapear los datos
- Leer y extraer datos sobre una ubicación específica
- Leer datos y crear un archivo csv como salida

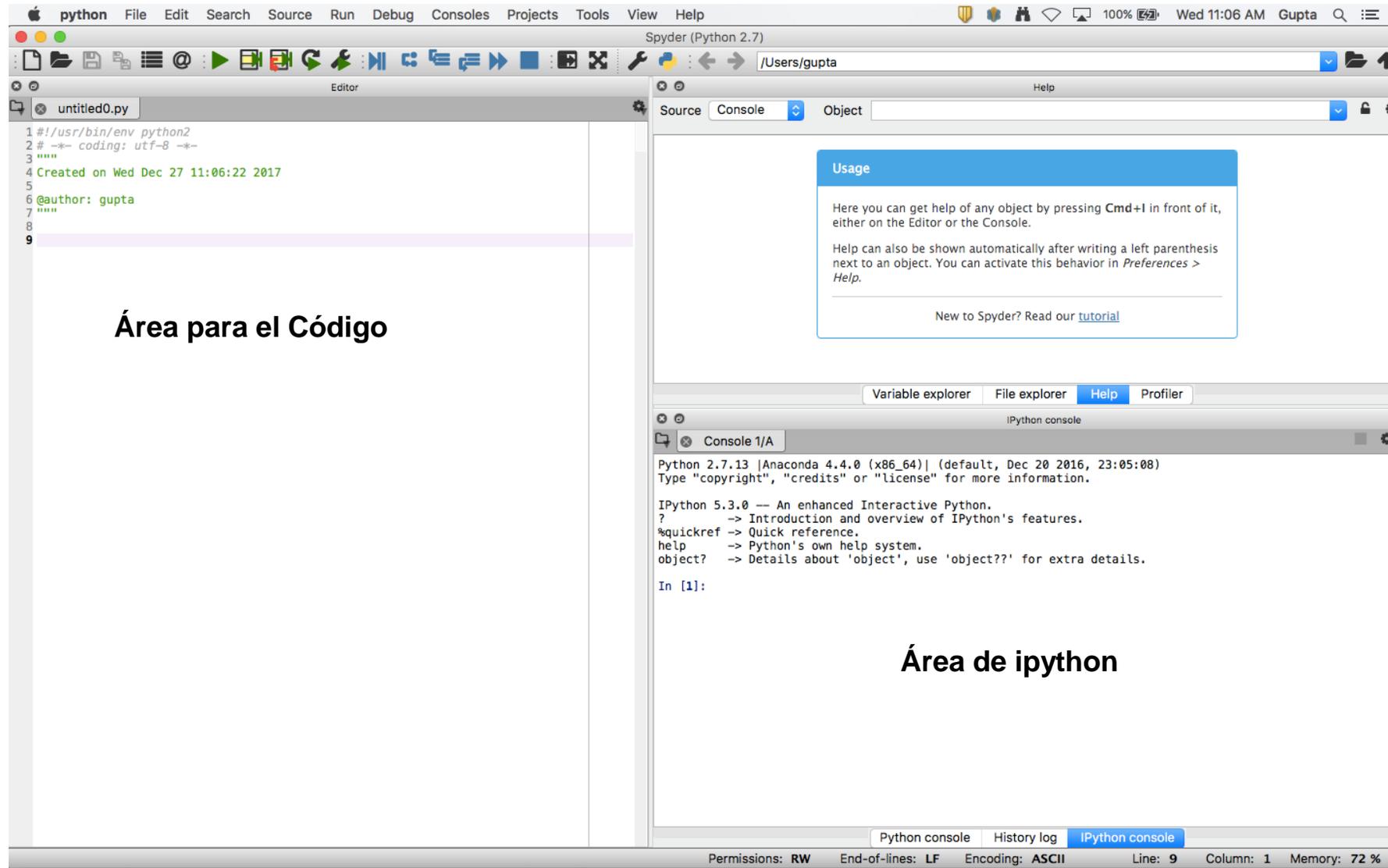
Datos y Códigos Necesarios

- Screenshot of ARSET page once material is posted

Anaconda y el Editor Spyder

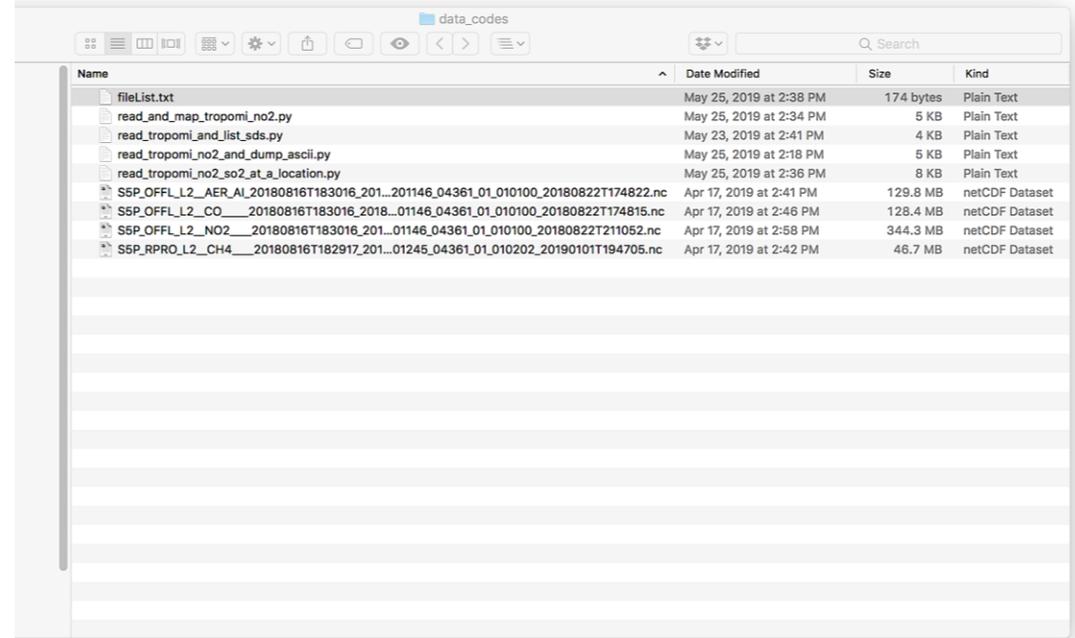


Vista de Spyder

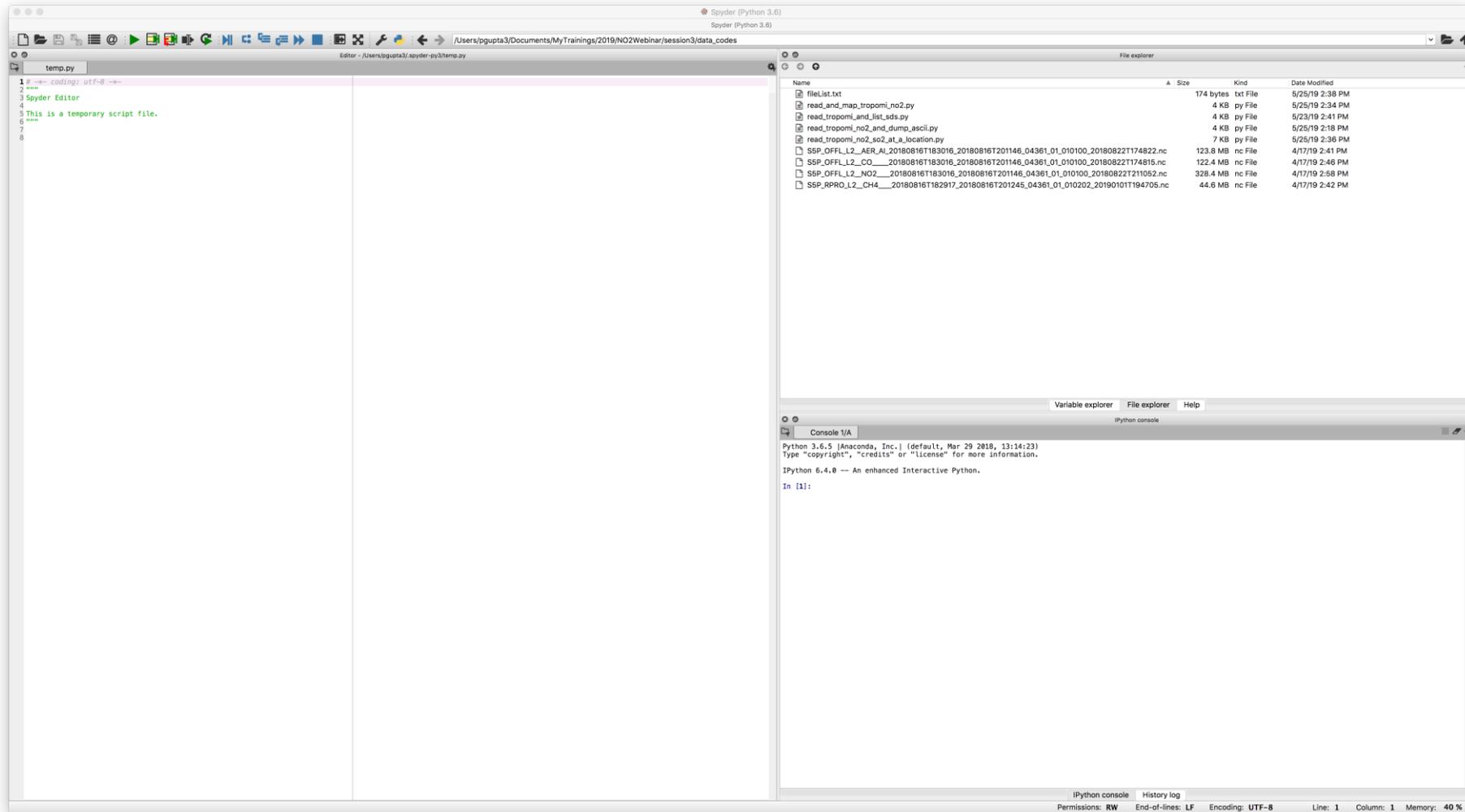


Vista del Directorio Actual y fileList.txt

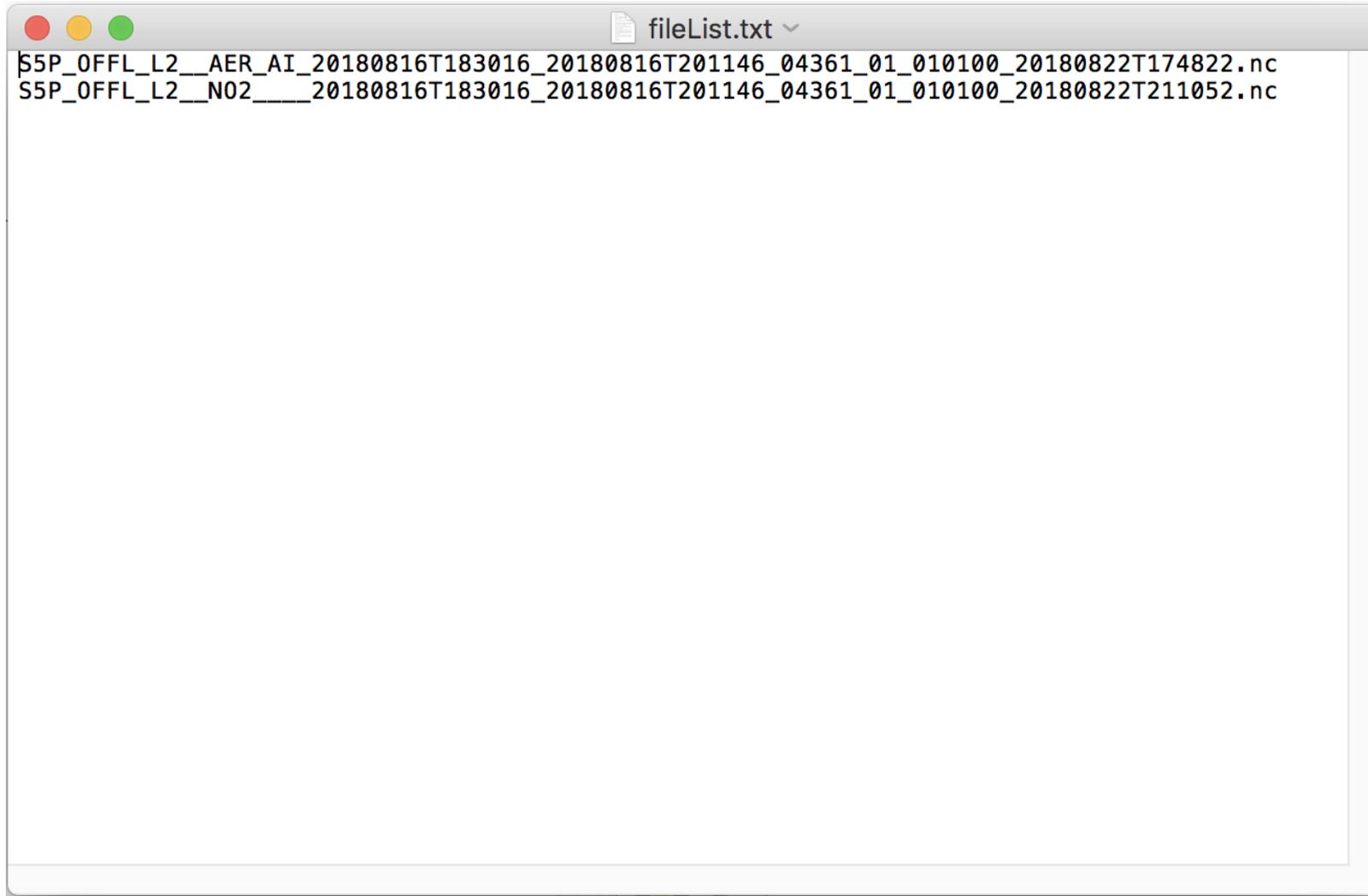
- En un archivo de texto, cree una lista de cada archivo netcdf de su interés y llámelo 'fileList.txt'
- El mismo directorio debería tener
 - Todos los códigos Python
 - Todos los archivos de datos netcdf (.nc)
 - Un archivo llamado 'fileList.txt' que contiene una lista del nombre de cada archivo netcdf



Vista de Spyder



fileList.txt



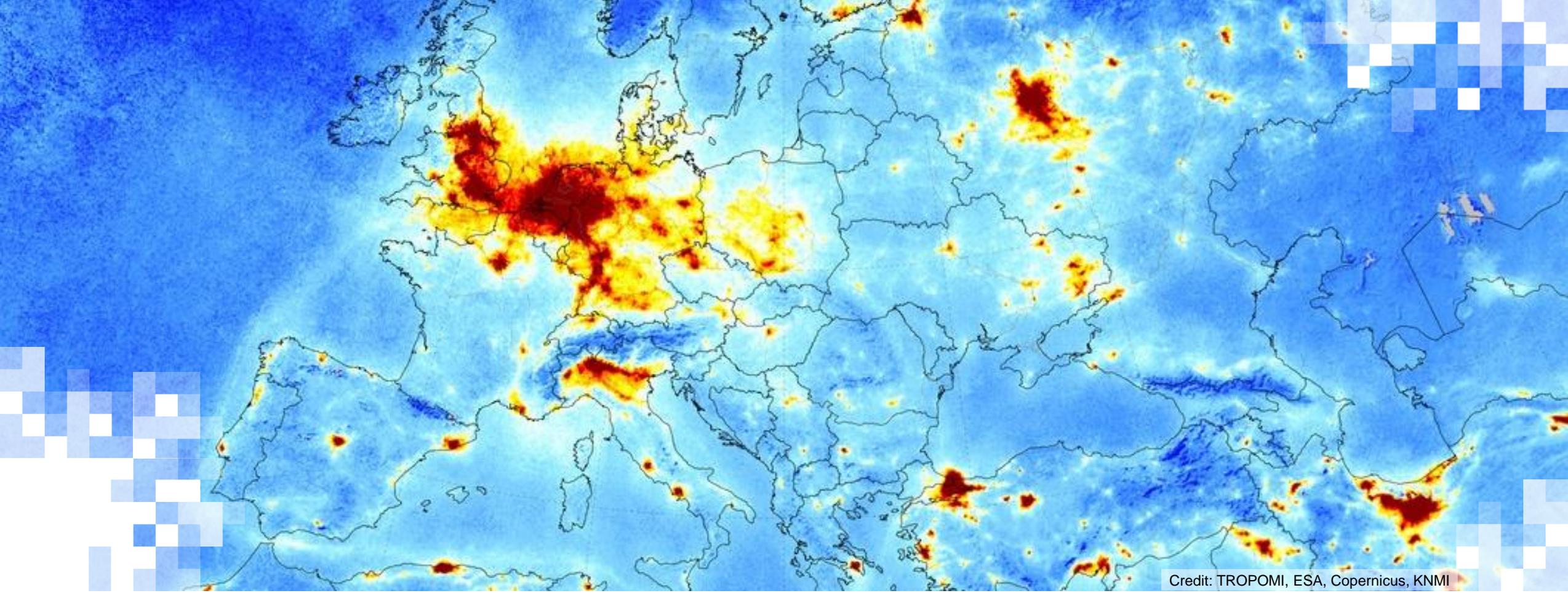
```
fileList.txt
$5P_OFFL_L2__AER_AI_20180816T183016_20180816T201146_04361_01_010100_20180822T174822.nc
$5P_OFFL_L2__N02____20180816T183016_20180816T201146_04361_01_010100_20180822T211052.nc
```

Paquetes de Python y Código de Prueba

Abra el código de prueba (test code) y ejecútelo

```
1#!/usr/bin/env python3
2# -*- coding: utf-8 -*-
3"""
4Created on Tue May 28 09:52:00 2019
5
6@author: pgupta3
7"""
8
9#!/usr/bin/python
10 from netCDF4 import Dataset
11 import numpy as np
12 from numpy import unravel_index
13 import sys
14 import time
15 import calendar
16 import datetime as dt
17 import pandas as pd
18 from mpl_toolkits.basemap import Basemap
19 import matplotlib.pyplot as plt
```

Si este código se ejecuta sin ningún error y genera una salida, significa que su Python está listo para la sesión de hoy.



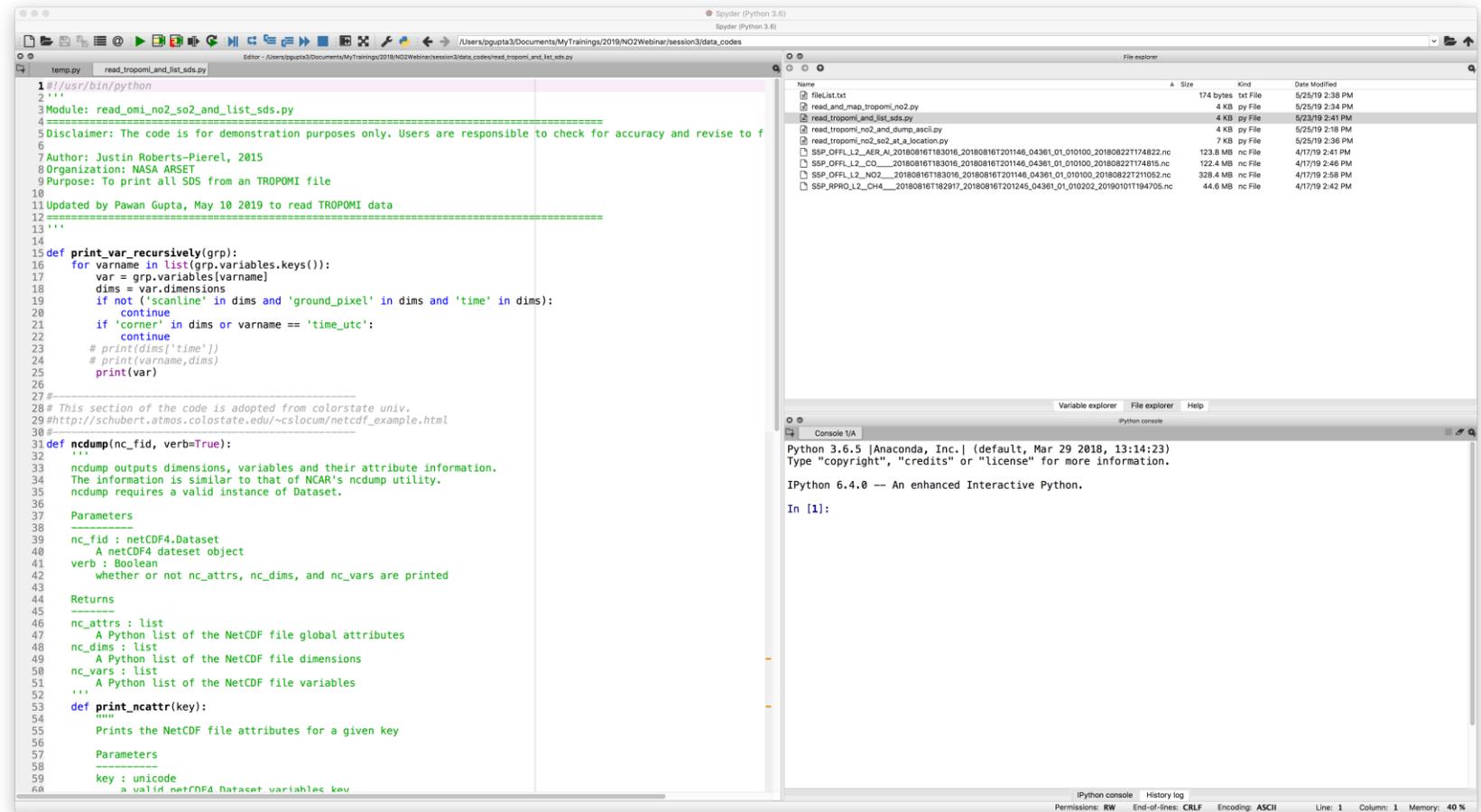
Leer un Archivo TROPOMI NO₂ (nc) e Imprimir Lista de SDS

Imprimir Conjuntos de Datos Científicos (Scientific Data Sets o SDSs)

read_tropomi_and_list_sds.py

Propósito: leer archivos TROPOMI de datos nivel 2 en formato netcdf e imprimir todos los **Conjuntos de Datos Científicos** (Scientific Data Sets o SDS).

En su forma actual, todos estos códigos sirven *únicamente para productos nivel 2, productos no cuadrículados*. Se prueba el código para datos NO₂ y puede que sea necesario modificarlo para que funcione con otros conjuntos de datos TROPOMI.



```
1 #!/usr/bin/python
2 '''
3 Module: read_omi_no2_so2_and_list_sds.py
4
5 Disclaimer: The code is for demonstration purposes only. Users are responsible to check for accuracy and revise to f
6
7 Author: Justin Roberts-Pierel, 2015
8 Organization: NASA ARSET
9 Purpose: To print all SDS from an TROPOMI file
10
11 Updated by Pawan Gupta, May 10 2019 to read TROPOMI data
12 =====
13 '''
14
15 def print_var_recursively(grp):
16     for varname in list(grp.variables.keys()):
17         var = grp.variables[varname]
18         dims = var.dimensions
19         if not ('scanline' in dims and 'ground_pixel' in dims and 'time' in dims):
20             continue
21         if 'corner' in dims or varname == 'time_utc':
22             continue
23         # print(dims['time'])
24         # print(varname,dims)
25         print(var)
26
27
28 # This section of the code is adopted from colorstate univ.
29 #http://schubert.atmos.colostate.edu/~cslocum/netcdf_example.html
30
31 def ncdump(nc_fid, verb=True):
32     '''
33     ncdump outputs dimensions, variables and their attribute information.
34     The information is similar to that of NCAR's ncdump utility.
35     ncdump requires a valid instance of Dataset.
36
37     Parameters
38     -----
39     nc_fid : netCDF4.Dataset
40         A netCDF4 dataset object
41     verb : Boolean
42         whether or not nc_attrs, nc_dims, and nc_vars are printed
43
44     Returns
45     -----
46     nc_attrs : list
47         A Python list of the NetCDF file global attributes
48     nc_dims : list
49         A Python list of the NetCDF file dimensions
50     nc_vars : list
51         A Python list of the NetCDF file variables
52     ... A Python list of the NetCDF file variables
53
54     def print_ncattr(key):
55         """
56         Prints the NetCDF file attributes for a given key
57
58         Parameters
59         -----
60         key : unicode
61             a valid netCDF4 Dataset variables key
```

Ejecución y Salida

- Haga clic en la flecha verde para ejecutar el código
- El código procesará todos los archivos en **fileList.txt** uno por uno
- Siga las instrucciones en la terminal **ipython** (es decir, teclee 'Y' o 'N' cuando se le pida y presione Enter)

The screenshot displays the Spyder Python IDE interface. The top toolbar contains a green play button, which is highlighted with a red box and a green arrow pointing to it from the first bullet point. The main editor window shows a Python script with a function `print_var_recursively` and a `ncdump` function. The console window at the bottom right, also highlighted with a red box, shows the output of the script. The output includes metadata for a NetCDF variable: `proposed_standard_name: ultraviolet_aerosol_index standard_error`, `comment: Precision of aerosol index from 388 and 354 nm`, `long_name: Precision of aerosol index from 388 and 354 nm`, `radiation_wavelength: [354, 388.]`, `coordinates: longitude latitude`, and `_FillValue: 9.96921e+36`. The console also shows the path `path = /PRODUCT` and the current shape `current shape = (1, 3245, 450)`. At the bottom of the console, it asks `Would you like to process SSP_OFFL_L2_NO2_20180816T183016_20180816T201146_04361_01_010100_20180822T211052.nc` and prompts for a response `(Y/N)`.

Salida



Editando el Código

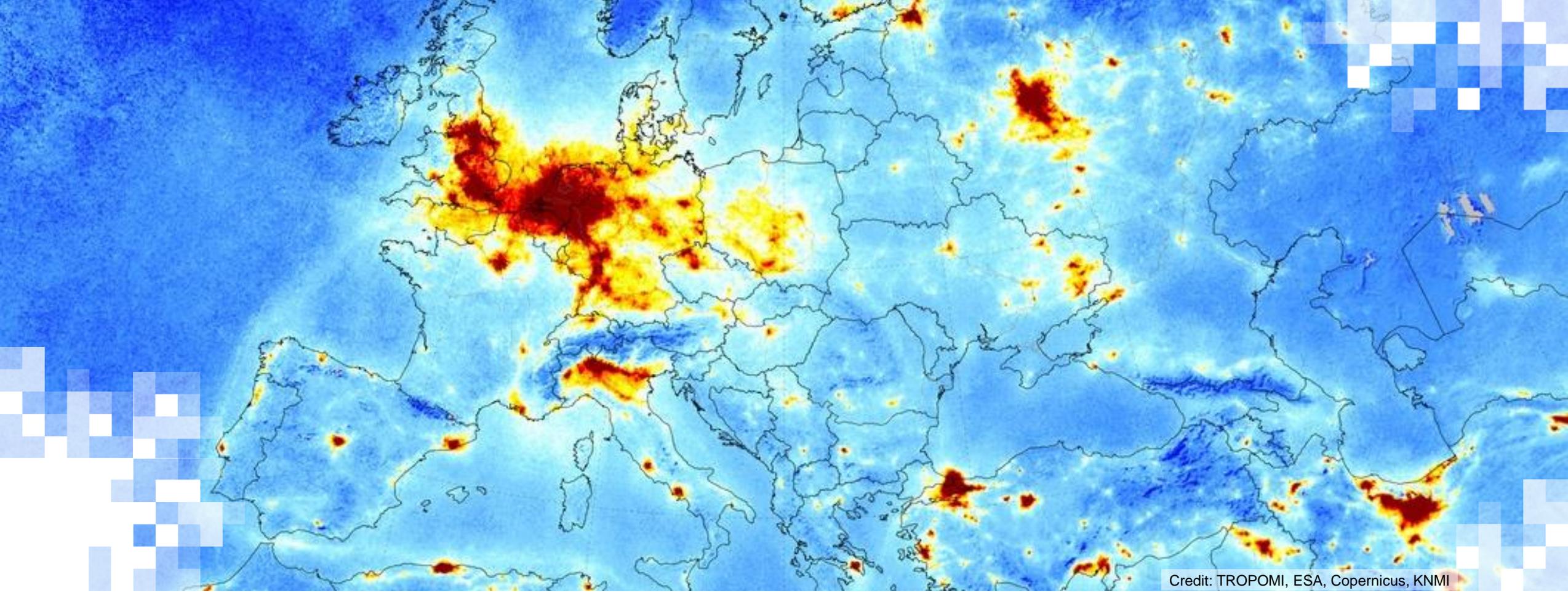
Cámbiele el nombre de fileList.txt a lo que usted quiera

```
95 #-----  
96 #import necessary modules  
97 import netCDF4  
98 from netCDF4 import Dataset  
99  
100 #This finds the user's current path so that all nc files can be found  
101 try:  
102     fileList=open('fileList.txt','r')  
103 except:  
104     print('Did not find a text file containing file names (perhaps name does not match)')  
105     sys.exit()  
106  
107 #loops through all files listed in the text file  
108 for FILE_NAME in fileList:  
109     FILE_NAME=FILE_NAME.strip()  
110     user_input=input('\nWould you like to process\n' + FILE_NAME + '\n\n(Y/N)')  
111     if(user_input == 'N' or user_input == 'n'):  
112         print('Skipping...')  
113         continue  
114     else:  
115         nc_file = Dataset(FILE_NAME, 'r') # 'r' means that nc file is open in read-only mode  
116         nc_attrs,nc_dims,nc_vars = ncdump(nc_file)  
117         print_var_recursively (nc_file.groups['PRODUCT'])  
118     nc_file.close()  
119
```

El grupo en TROPOMI donde los datos están almacenados se llama 'PRODUCT'. Hay otros grupos también en el archivo de datos.

Aplicaciones

- Los datos de NO₂ TROPOMI Nivel 2 y otros están en formato netCDF (.nc)
- Cada archivo nc contiene varios parámetros geofísicos
- Se necesitan códigos/herramientas especiales para abrir los archivos nc
- Este código ayuda a los usuarios a ver los nombres de los SDSs disponibles dentro de un archivo nc para análisis adicional



Credit: TROPOMI, ESA, Copernicus, KNMI

Mapear NO₂

Recordatorios

- Cerrar el código anterior en Spyder
- Reiniciar el kernel (núcleo) de ipython

Diagramar y Guardar un Mapa de TROPOMI AI* y NO₂

read_and_map_tropomi_no2_ai.py

```
1#!/usr/bin/python
2'''
3Module: read_and_map_tropomi_no2.py
4=====
5Disclaimer: The code is for demonstration purposes only. Users are responsible to check for accuracy and revise to f
6
7Author: Justin Roberts-Pierel, 2015
8Organization: NASA ARSET
9Purpose: To extract No2 or SO2 data from an OMI HDF5 file and create a map of the resulting data
10
11See the README associated with this module for more information.
12Modified by Vikalp Mishra & Pawan Gupta, May 10 2019 to read TROPOMI data
13'''
14'''
15import numpy as np
16from mpl_toolkits.basemap import Basemap
17import matplotlib.pyplot as plt
18import sys
19from netCDF4 import Dataset
20
21#This finds the user's current path so that all hdf4 files can be found
22try:
23     fileList=open('fileList.txt','r')
24except:
25     print('Did not find a text file containing file names (perhaps name does not match)')
26     sys.exit()
27
28#loops through all files listed in the text file
29for FILE_NAME in fileList:
30     FILE_NAME=FILE_NAME.strip()
31     user_input=input('\nWould you like to process\n' + FILE_NAME + '\n\n(Y/N)')
32     if(user_input == 'N' or user_input == 'n'):
33         print('Skipping...')
34         continue
35     else:
36         file = Dataset(FILE_NAME, 'r')
37# read the data
38     ds=file
39     grp='PRODUCT'
40     lat= ds.groups[grp].variables['latitude'][0][:][:]
41     lon= ds.groups[grp].variables['longitude'][0][:][:]
42     if 'NO2' in FILE_NAME:
43         sds_name='nitrogen_dioxide_tropospheric_column'
44     if 'AER_AI' in FILE_NAME:
45         sds_name='aerosol_index_354_388'
46     data= ds.groups[grp].variables[sds_name]
47
48     #get necessary attributes
49     fv=data._FillValue
50
51     #get lat and lon information
52     min_lat=np.min(lat)
53     max_lat=np.max(lat)
54     min_lon=np.min(lon)
55     max_lon=np.max(lon)
56
57     # set map labels
58     map_label = data.units
59     map_title = data.long_name
60
```

Name	Size	Kind	Date Modified
fileList.txt	174 bytes	txt File	5/25/19 2:38 PM
read_and_map_tropomi_no2.py	4 KB	py File	5/25/19 2:34 PM
read_tropomi_and_list_sds.py	4 KB	py File	5/23/19 2:41 PM
read_tropomi_no2_and_dump_ascii.py	4 KB	py File	5/25/19 2:18 PM
read_tropomi_no2_so2_at_a_location.py	7 KB	py File	5/25/19 2:36 PM
SSP_OFFL_L2_AER_AI_20180816T183016_20180816T201146_04361_01_010100_20180822T174822.nc	123.8 MB	nc File	4/17/19 2:41 PM
SSP_OFFL_L2_CO_20180816T183016_20180816T201146_04361_01_010100_20180822T174815.nc	122.4 MB	nc File	4/17/19 2:46 PM
SSP_OFFL_L2_NO2_20180816T183016_20180816T201146_04361_01_010100_20180822T171052.nc	328.4 MB	nc File	4/17/19 2:58 PM
SSP_RPRO_L2_CH4_20180816T182917_20180816T201245_04361_01_010202_20190101T194705.nc	44.6 MB	nc File	4/17/19 2:42 PM

Python 3.6.5 |Anaconda, Inc.| (default, Mar 29 2018, 13:14:23)
Type "copyright", "credits" or "license" for more information.
IPython 6.4.0 -- An enhanced Interactive Python.
Restarting kernel...

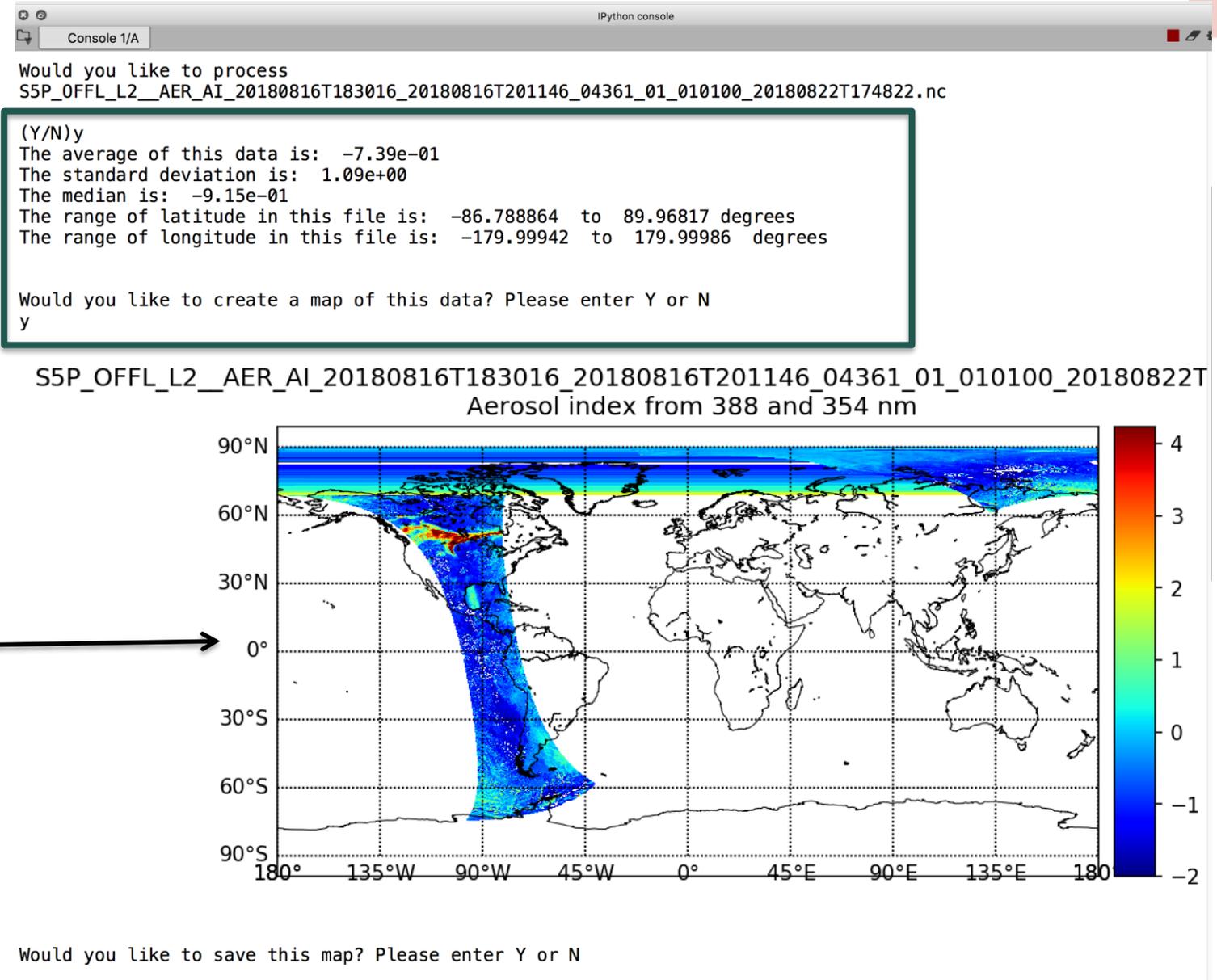
In [1]:

*AI= siglas de “Aerosol Index”, Índice de Aerosoles, en inglés

[NASA's Applied Remote Sensing Training Program](#)

Ejecución y Salida

Estadísticas del
AI/NO₂

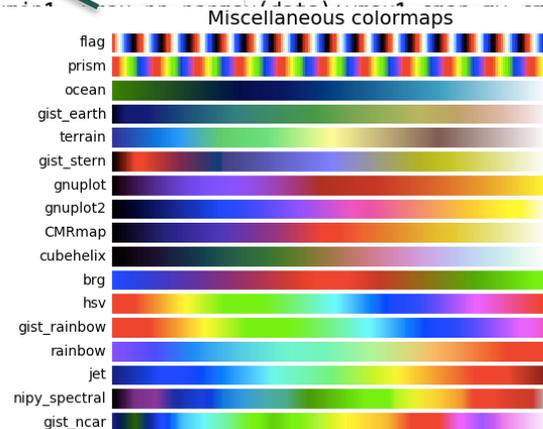


Mapa de salida

Editando el Código

Cambie la escala cromática

```
#if user would like a map, view it
if is_map == 'Y' or is_map == 'y':
    data = np.ma.masked_array(data, np.isnan(data))
    m = Basemap(projection='cyl', resolution='l',
                llcrnrlat=-90, urcrnrlat = 90,
                llcrnrlon=-180, urcrnrlon = 180)
    m.drawcoastlines(linewidth=0.5)
    m.drawparallels(np.arange(-90., 120., 30.), labels=[1, 0, 0, 0])
    m.drawmeridians(np.arange(-180, 180., 45.), labels=[0, 0, 0, 1])
    my_cmap = plt.cm.get_cmap('jet')
    my_cmap.set_under('w')
    vmin1=0.0
    vmax1=0.05
    if 'AER_AI' in FILE_NAME:
        vmin1=-2.0
        vmax1=0.4
    m.pcolormesh(lon, lat, data, latlon=True, vmin=vmin1, vmax=vmax1)
    cb = m.colorbar()
    cb.set_label(map_label)
    plt.autoscale()
    #title the plot
    plt.title('{0}\n {1}'.format(FILE_NAME, map_title))
    fig = plt.gcf()
    # Show the plot window.
    plt.show()
#once you close the map it asks if you'd like to save it
is_save=str(input('\nWould you like to save this map? (y/n)'))
if is_save == 'Y' or is_save == 'y':
    #saves as a png if the user would like
    pngfile = '{0}.png'.format(FILE_NAME[:-3])
    fig.savefig(pngfile, dpi = 300)
#close the hdf5 file
file.close()
```



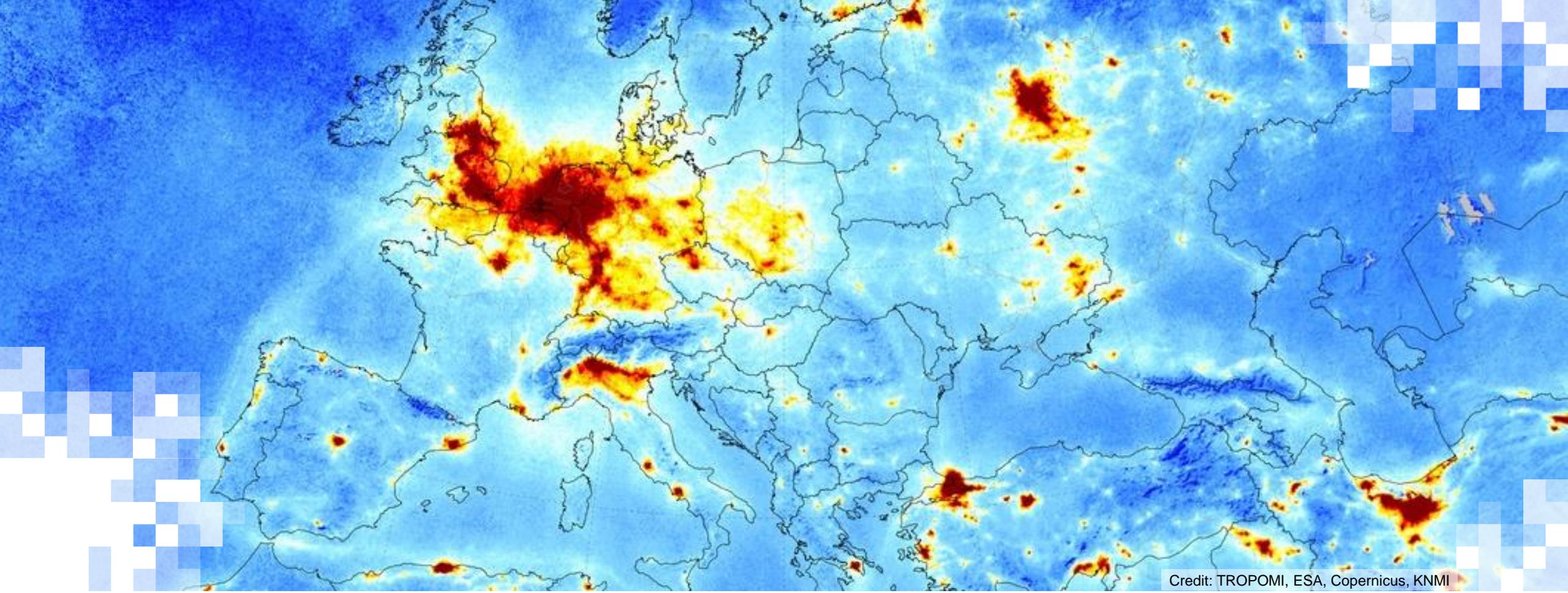
Cambie el SDS a diagramar

```
37 # read the data
38 ds=file
39 grp='PRODUCT'
40 lat= ds.groups[grp].variables['latitude'][0][:][:]
41 lon= ds.groups[grp].variables['longitude'][0][:][:]
42 if 'NO2' in FILE_NAME:
43     sds_name='nitrogendioxide_tropospheric_column'
44 if 'AER_AI' in FILE_NAME:
45     sds_name='aerosol_index_354_388'
46 data= ds.groups[grp].variables[sds_name]
47
```

https://matplotlib.org/examples/color/colormaps_reference.html

Aplicaciones

- Este es un ejemplo de código para leer y mapear datos de NO₂ y AI Nivel 2 de TROPOMI
- Se puede modificar el código para diferentes necesidades cartográficas
- El usuario puede crear mapas diarios de columnas de gases trazadores sobre ciertas regiones y empezar a analizar cambios a través del tiempo
- Estos mapas también pueden ayudar a identificar regiones de alta contaminación



Extraer NO_2/AI en una Ubicación Particular

Extraer Valores de NO₂ de Datos TROPOMI Nivel 2

read_tropomi_no2_ai_at_a_location.py

- **Propósito:** leer un archivo de datos de NO₂/AI TROPOMI nivel 2 en formato nc y extraer valores en un punto particular en el suelo

```
13 '''
14
15 #import necessary modules
16 import numpy as np
17 import sys
18 from numpy import unravel_index
19 from netCDF4 import Dataset
20
21 #This finds the user's current path so that all hdf4 files can be found
22 try:
23     fileList=open('fileList.txt','r')
24 except:
25     print('Did not find a text file containing file names (perhaps name does not match)')
26     sys.exit()
27
28 #loops through all files listed in the text file
29 for FILE_NAME in fileList:
30     FILE_NAME=FILE_NAME.strip()
31     user_input=input('\nWould you like to process\n' + FILE_NAME + '\n\n(Y/N)')
32     if(user_input == 'N' or user_input == 'n'):
33         print('Skipping...')
34         continue
35     else:
36         file = Dataset(FILE_NAME, 'r')
37 # read the data
38         if 'NO2' in FILE_NAME:
39             print('This is an TROPOMI NO2 file.')
40             #this is how you access the data tree in an hdf5 file
41             SDS_NAME='nitrogen_dioxide_tropospheric_column'
42         elif 'AER_AI' in FILE_NAME:
43             print('This is an TROPOMI Aerosol Index file.')
44             SDS_NAME='aerosol_index_354_388'
45         ds=file
46         grp='PRODUCT'
47         lat= ds.groups[grp].variables['latitude'][0][:]
48         lon= ds.groups[grp].variables['longitude'][0][:]
49         data= ds.groups[grp].variables[SDS_NAME]
50
51 #get necessary attributes
52 fv=data._FillValue
53
54 #get lat and lon information
55 min_lat=np.min(lat)
56 max_lat=np.max(lat)
57 min_lon=np.min(lon)
58 max_lon=np.max(lon)
59
60 # set map labels
61 map_label = data.units
62 map_title = data.long_name
63 SDS_NAME=map_title
64
65 #get the data as an array and mask fill/missing values
66 dataArray=np.array(data[0][:])
67 dataArray[dataArray==fv]=np.nan
68 data=dataArray
69
70 #get statistics about data
71 average=np.nanmean(dataArray)
72 stdev=np.nanstd(dataArray)
```

File explorer

Name	Size	Kind	Date Modified
fileList.txt	174 bytes	txt File	5/25/19 2:38 PM
read_and_map_tropomi_no2.py	4 KB	py File	5/25/19 2:34 PM
read_tropomi_and_list_sds.py	4 KB	py File	5/23/19 2:41 PM
read_tropomi_no2_and_dump_asci.py	4 KB	py File	5/25/19 2:18 PM
read_tropomi_no2_so2_at_a_location.py	7 KB	py File	5/25/19 2:36 PM
SSP_OFFL_L2_AER_AI_20180816T183016_20180816T201146_04361_01_010100_20180822T174822.nc	123.8 MB	nc File	4/17/19 2:41 PM
SSP_OFFL_L2_CO_20180816T183016_20180816T201146_04361_01_010100_20180822T174815.nc	122.4 MB	nc File	4/17/19 2:46 PM
SSP_OFFL_L2_NO2_20180816T183016_20180816T201146_04361_01_010100_20180822T110552.nc	328.4 MB	nc File	4/17/19 2:58 PM
SSP_RPRO_L2_CH4_20180816T182917_20180816T201245_04361_01_010202_20190101T194705.nc	44.6 MB	nc File	4/17/19 2:42 PM

Console I/A

```
Restarting kernel...

In [1]: runfile('/Users/pgupta3/Documents/MyTrainings/2019/NO2Webinar/session3/data_codes/
read_tropomi_no2_so2_at_a_location.py', wdir='/Users/pgupta3/Documents/MyTrainings/2019/NO2Webinar/session3/
data_codes')

Would you like to process
SSP_OFFL_L2_AER_AI_20180816T183016_20180816T201146_04361_01_010100_20180822T174822.nc

(Y/N)y
This is an TROPOMI Aerosol Index file.
The average of this data is: -7.39e-01
The standard deviation is: 1.09e+00
The median is: -9.15e-01
The range of latitude in this file is: -86.788864 to 89.96817 degrees
The range of longitude in this file is: -179.99942 to 179.99986 degrees

Please enter the latitude you would like to analyze (Deg. N): 35.5
Please enter the longitude you would like to analyze (Deg. E): -100.0

The nearest pixel to your entered location is at:
Latitude: 35.51131 Longitude: -100.01028
The value of Aerosol index from 388 and 354 nm at this pixel is -2.39e+00
There are 9 valid pixels in a 3x3 grid centered at your entered location.
The average value in this grid is: -2.07e+00
The median value in this grid is: -2.04e+00
The standard deviation in this grid is: 3.23e-01

There are 25 valid pixels in a 5x5 grid centered at your entered location.

The average value in this grid is: -1.83e+00
The median value in this grid is: -1.92e+00
The standard deviation in this grid is: 4.55e-01

Would you like to process
SSP_OFFL_L2_NO2_20180816T183016_20180816T201146_04361_01_010100_20180822T110552.nc

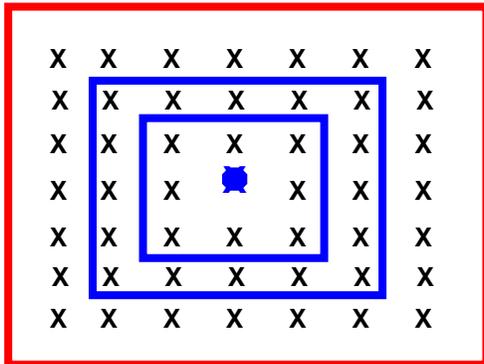
(Y/N)
```

Ejecución y Salida

Teclee "Y" para procesar el archivo, "N" para saltar

Lat. y Lon. de la estación

Salidas



```
Would you like to process
S5P_OFFL_L2_N02____20180816T183016_20180816T201146_04361_01_010100_20180822T211052.nc
```

```
(Y/N)y
This is an TROPOMI N02 file.
The average of this data is: 1.13e-06
The standard deviation is: 2.96e-05
The median is: 4.12e-06
The range of latitude in this file is: -86.788864 to 89.96817 degrees
The range of longitude in this file is: -179.99942 to 179.99986 degrees
```

```
Please enter the latitude you would like to analyze (Deg. N): 35.0
Please enter the longitude you would like to analyze (Deg. E): -100.0
```

```
The nearest pixel to your entered location is at:
Latitude: 35.02769 Longitude: -99.99893
The value of Tropospheric vertical column of nitrogen dioxide at this pixel is 2.23e-05
There are 9 valid pixels in a 3x3 grid centered at your entered location.
The average value in this grid is: 2.15e-05
The median value in this grid is: 2.13e-05
The standard deviation in this grid is: 5.89e-06
```

```
There are 25 valid pixels in a 5x5 grid centered at your entered location.
```

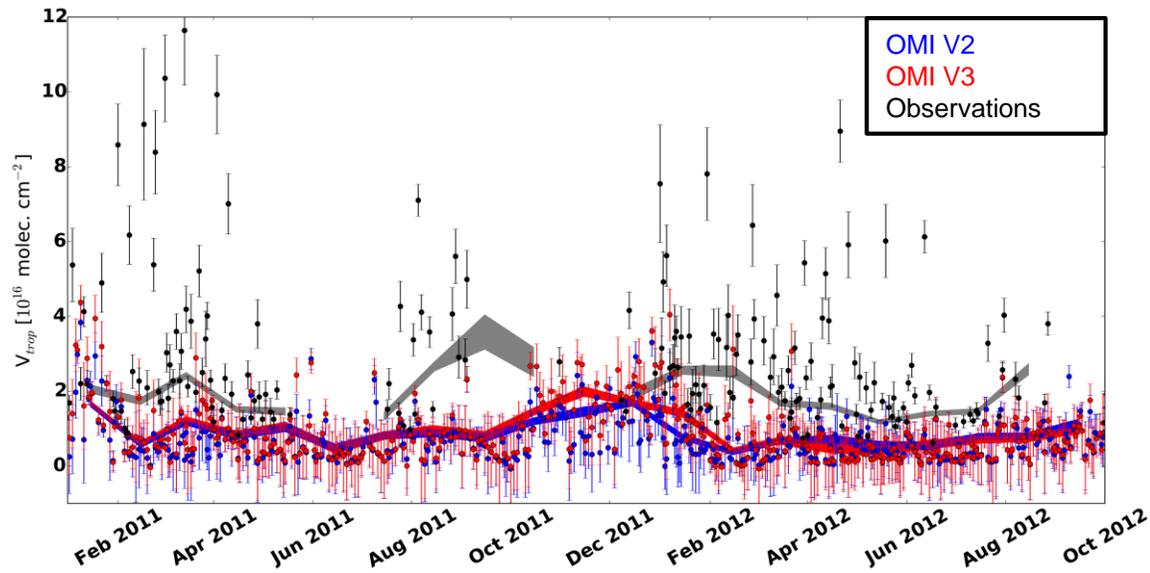
```
The average value in this grid is: 2.85e-05
The median value in this grid is: 2.60e-05
The standard deviation in this grid is: 9.31e-06
```

Editando el Código – Cambie de SDS

```
27
28 #loops through all files listed in the text file
29 for FILE_NAME in fileList:
30     FILE_NAME=FILE_NAME.strip()
31     user_input=input('\nWould you like to process\n' + FILE_NAME + '\n\n(Y/N)')
32     if(user_input == 'N' or user_input == 'n'):
33         print('Skipping...')
34         continue
35     else:
36         file = Dataset(FILE_NAME, 'r')
37 # read the data
38         if 'NO2' in FILE_NAME:
39             print('This is an TROPOMI NO2 file.')
40             #this is how you access the data tree in an hdf5 file
41             SDS_NAME='nitrogen_dioxide_tropospheric_column'
42         elif 'AER_AI' in FILE_NAME:
43             print('This is an TROPOMI Aerosol Index file.')
44             SDS_NAME='aerosol_index_354_388'
45         ds=file
46         grp='PRODUCT'
47         lat= ds.groups[grp].variables['latitude'][0][:][:]
48         lon= ds.groups[grp].variables['longitude'][0][:][:]
49         data= ds.groups[grp].variables[SDS_NAME]
50
51         #get necessary attributes
52         fv=data._FillValue
53
```

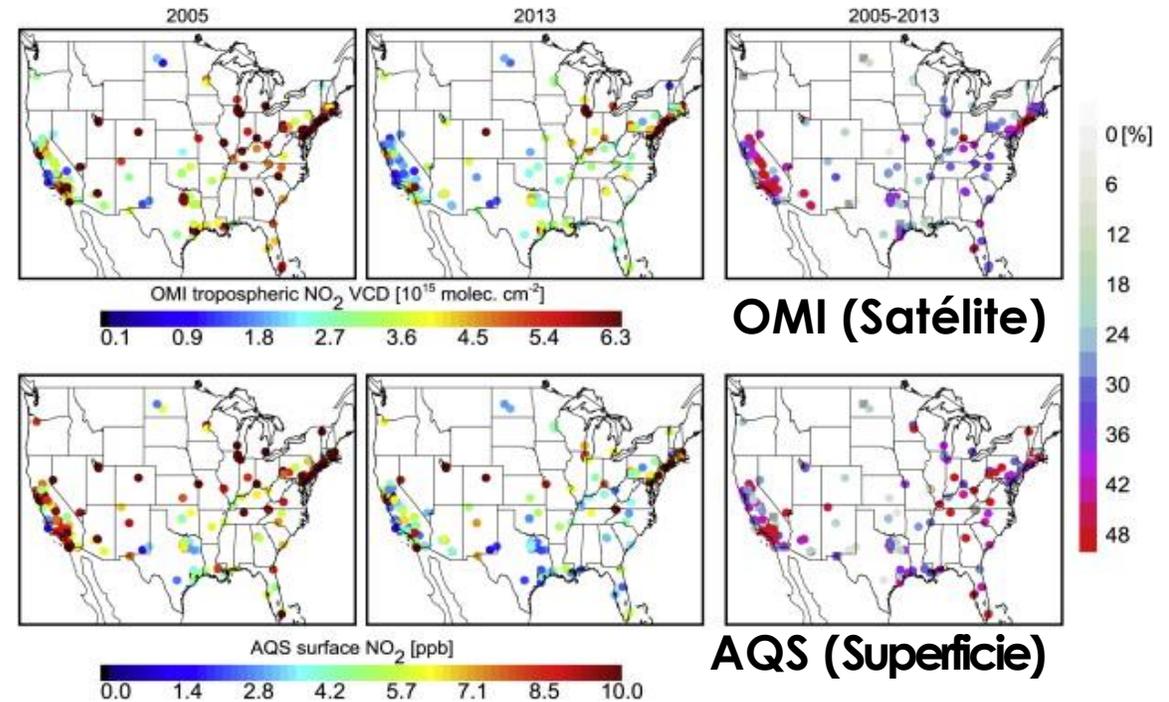
Aplicaciones

Validación por Satélite

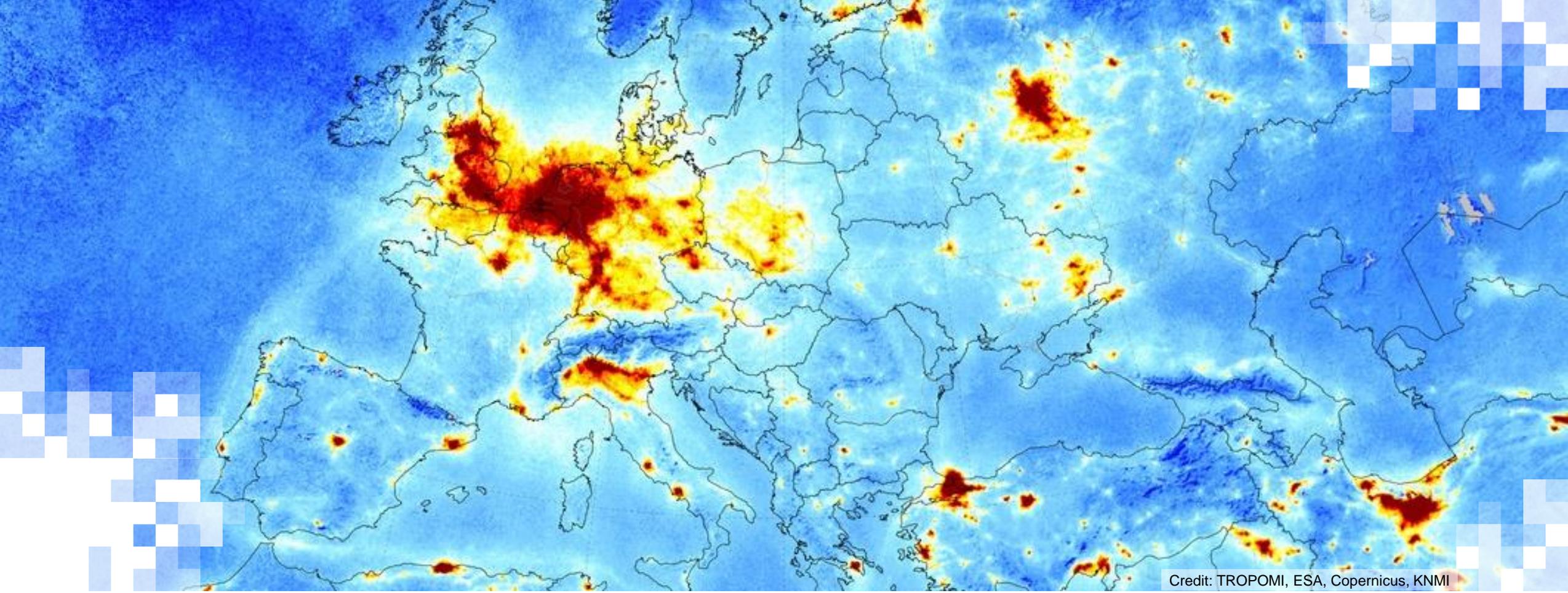


Fuente: Krotkov et al. (2017)

Columna vs. Superficie Relaciones y Tendencias



Fuente: Lamsal, L.N. et al. (2016)



Exportar Variables nc a Formato CSV

Exportar Variables nc de TROPOMI NO₂/AI Como Salidas a un Archivo CSV

read_tropomi_no2_ai_and_dump_ascii.py

- **Propósito:** leer un archivo de datos TROPOMI de NO₂ o AI nivel 2 en formato netCDF y escribir ciertos SDSs en un archivo csv (texto)

The screenshot displays the Spyder Python IDE interface. The main editor window shows a Python script named `read_tropomi_no2_and_dump_ascii.py`. The script includes a disclaimer, author information (Justin Roberts-Pierele, 2015), and a purpose statement: "To print all SDS from a TROPOMI file". The code uses `netCDF4` to read data from a file list, processes TROPOMI NO₂ and AI files, and exports specific SDSs to CSV files. The console window shows the execution output, including the file path, the file name, and the list of variables being saved to the CSV file.

```
1#!/usr/bin/python
2'''
3Module: read_tropomi_no2_and_dump_ascii.py
4
5Disclaimer: The code is for demonstration purposes only. Users are responsible to check for accuracy and revise to f
6
7Author: Justin Roberts-Pierele, 2015
8Organization: NASA ARSET
9Purpose: To print all SDS from a TROPOMI file
10
11Modified by Vikalp Mishra & Pawan Gupta, May 10 2019 to read TROPOMI data
12'''
13'''
14#!/usr/bin/python
15from netCDF4 import Dataset
16import numpy as np
17import sys
18import time
19import calendar
20import datetime as dt
21import pandas as pd
22
23
24#This finds the user's current path so that all hdf4 files can be found
25try:
26    fileList=open('fileList.txt','r')
27except:
28    print('Did not find a text file containing file names (perhaps name does not match)')
29    sys.exit()
30
31#loops through all files listed in the text file
32for FILE_NAME in fileList:
33    FILE_NAME=FILE_NAME.strip()
34    user_input=input('\nWould you like to process\n' + FILE_NAME + '\n\n(Y/N)')
35    if(user_input == 'N' or user_input == 'n'):
36        print('Skipping...')
37        continue
38    else:
39        file = Dataset(FILE_NAME, 'r')
40        grp='PRODUCT'
41# read the data
42        if 'NO2' in FILE_NAME:
43            print('This is an TROPOMI NO2 file.')
44            #this is how you access the data tree in an hdf5 file
45            SDS_NAME='nitrogen_dioxide_tropospheric_column'
46        elif 'AER_AI' in FILE_NAME:
47            print('This is an TROPOMI Aerosol Index file.')
48            SDS_NAME='aerosol_index_354_388'
49        ds=file
50        grp='PRODUCT'
51        lat= ds.groups[grp].variables['latitude'][0][:][:]
52        lon= ds.groups[grp].variables['longitude'][0][:][:]
53        data= ds.groups[grp].variables[SDS_NAME]
54
55# get necessary attributes
56        fv=data._FillValue
57
58# get scan time and turn it into a vector
59        scan_time= ds.groups[grp].variables['time_utc']
60        # scan_time=nonlinear_interp(scan_time[1:-1], scan_time[0])
```

File explorer shows a list of files, including `SSP_OFFL_L2_AER_AI_20180816T183016_20180816T201146_04361_01_010100_20180822T174822.csv`.

Python console output:

```
Python 3.6.5 [Anaconda, Inc.] (default, Mar 29 2018, 13:14:23)
Type "copyright", "credits" or "license" for more information.

IPython 6.4.0 -- An enhanced Interactive Python.

Restarting kernel...

In [1]: runfile('/Users/pgupta3/Documents/MyTrainings/2019/NO2Webinar/session3/data_codes/
read_tropomi_no2_and_dump_ascii.py', wdir='/Users/pgupta3/Documents/MyTrainings/2019/NO2Webinar/session3/
data_codes')

Would you like to process
SSP_OFFL_L2_AER_AI_20180816T183016_20180816T201146_04361_01_010100_20180822T174822.nc

(Y/N)y
This is an TROPOMI Aerosol Index file.
latitude (1, 3245, 450)
longitude (1, 3245, 450)
qa_value (1, 3245, 450)
aerosol_index_354_388 (1, 3245, 450)
aerosol_index_340_388 (1, 3245, 450)
aerosol_index_354_388_precision (1, 3245, 450)
aerosol_index_340_388_precision (1, 3245, 450)

All files have been saved successfully.

Would you like to process
SSP_OFFL_L2_NO2_20180816T183016_20180816T201146_04361_01_010100_20180822T11052.nc

(Y/N)y
This is an TROPOMI NO2 file.
latitude (1, 3245, 450)
longitude (1, 3245, 450)
qa_value (1, 3245, 450)
nitrogen_dioxide_tropospheric_column (1, 3245, 450)
nitrogen_dioxide_tropospheric_column_precision (1, 3245, 450)
nitrogen_dioxide_tropospheric_column_precision_kernel (1, 3245, 450)
air_mass_factor_troposphere (1, 3245, 450)
air_mass_factor_total (1, 3245, 450)
tm5_tropopause_layer_index (1, 3245, 450)
```

Salida

The screenshot shows an Excel spreadsheet with the following data structure:

Year	Month	Day	Hour	Minute	Second	latitude	longitude	qa_value	nitrogendiox	nitrogendiox	nitrogendiox	air_mass_fa	air_mass_fa	tm5_tropopause_layer_index
2018	51	16	18	51	51	-85.283333	-136.46013	0	9.97E+36	9.97E+36	9.97E+36	9.97E+36	9.97E+36	-2.147E+09
2018	51	16	18	51	51	-85.347733	-135.71352	0	9.97E+36	9.97E+36	9.97E+36	9.97E+36	9.97E+36	-2.147E+09
2018	51	16	18	51	51	-85.410431	-134.95714	0	9.97E+36	9.97E+36	9.97E+36	9.97E+36	9.97E+36	-2.147E+09
2018	51	16	18	51	51	-85.471458	-134.19095	0	9.97E+36	9.97E+36	9.97E+36	9.97E+36	9.97E+36	-2.147E+09
2018	51	16	18	51	51	-85.530846	-133.41499	0	9.97E+36	9.97E+36	9.97E+36	9.97E+36	9.97E+36	-2.147E+09
2018	51	16	18	51	51	-85.588608	-132.6293	0	9.97E+36	9.97E+36	9.97E+36	9.97E+36	9.97E+36	-2.147E+09
2018	51	16	18	51	51	-85.644768	-131.83389	0	9.97E+36	9.97E+36	9.97E+36	9.97E+36	9.97E+36	-2.147E+09
2018	51	16	18	51	51	-85.699348	-131.02884	0	9.97E+36	9.97E+36	9.97E+36	9.97E+36	9.97E+36	-2.147E+09
2018	51	16	18	51	51	-85.752373	-130.21422	0	9.97E+36	9.97E+36	9.97E+36	9.97E+36	9.97E+36	-2.147E+09
2018	51	16	18	51	51	-85.803864	-129.39012	0	9.97E+36	9.97E+36	9.97E+36	9.97E+36	9.97E+36	-2.147E+09
2018	51	16	18	51	51	-85.853828	-128.55666	0	9.97E+36	9.97E+36	9.97E+36	9.97E+36	9.97E+36	-2.147E+09
2018	51	16	18	51	51	-85.902298	-127.71396	0	9.97E+36	9.97E+36	9.97E+36	9.97E+36	9.97E+36	-2.147E+09
2018	51	16	18	51	51	-85.94928	-126.86217	0	9.97E+36	9.97E+36	9.97E+36	9.97E+36	9.97E+36	-2.147E+09
2018	51	16	18	51	51	-85.994797	-126.00148	0	9.97E+36	9.97E+36	9.97E+36	9.97E+36	9.97E+36	-2.147E+09
2018	51	16	18	51	51	-86.038864	-125.13205	0	9.97E+36	9.97E+36	9.97E+36	9.97E+36	9.97E+36	-2.147E+09
2018	51	16	18	51	51	-86.081497	-124.2541	0	9.97E+36	9.97E+36	9.97E+36	9.97E+36	9.97E+36	-2.147E+09
2018	51	16	18	51	51	-86.122704	-123.36785	0	9.97E+36	9.97E+36	9.97E+36	9.97E+36	9.97E+36	-2.147E+09
2018	51	16	18	51	51	-86.162514	-122.47355	0	9.97E+36	9.97E+36	9.97E+36	9.97E+36	9.97E+36	-2.147E+09
2018	51	16	18	51	51	-86.200935	-121.57148	0	9.97E+36	9.97E+36	9.97E+36	9.97E+36	9.97E+36	-2.147E+09
2018	51	16	18	51	51	-86.237976	-120.66191	0	9.97E+36	9.97E+36	9.97E+36	9.97E+36	9.97E+36	-2.147E+09
2018	51	16	18	51	51	-86.273666	-119.74516	0	9.97E+36	9.97E+36	9.97E+36	9.97E+36	9.97E+36	-2.147E+09
2018	51	16	18	51	51	-86.307999	-118.82155	0	9.97E+36	9.97E+36	9.97E+36	9.97E+36	9.97E+36	-2.147E+09
2018	51	16	18	51	51	-86.357018	-117.42407	0	9.97E+36	9.97E+36	9.97E+36	9.97E+36	9.97E+36	-2.147E+09
2018	51	16	18	51	51	-86.417801	-115.54012	0	9.97E+36	9.97E+36	9.97E+36	9.97E+36	9.97E+36	-2.147E+09
2018	51	16	18	51	51	-86.47345	-113.63558	0	9.97E+36	9.97E+36	9.97E+36	9.97E+36	9.97E+36	-2.147E+09
2018	51	16	18	51	51	-86.524094	-111.71391	0	9.97E+36	9.97E+36	9.97E+36	9.97E+36	9.97E+36	-2.147E+09
2018	51	16	18	51	51	-86.569855	-109.77879	0	9.97E+36	9.97E+36	9.97E+36	9.97E+36	9.97E+36	-2.147E+09
2018	51	16	18	51	51	-86.61084	-107.83408	0	9.97E+36	9.97E+36	9.97E+36	9.97E+36	9.97E+36	-2.147E+09
2018	51	16	18	51	51	-86.647194	-105.88378	0	9.97E+36	9.97E+36	9.97E+36	9.97E+36	9.97E+36	-2.147E+09
2018	51	16	18	51	51	-86.679039	-103.93198	0	9.97E+36	9.97E+36	9.97E+36	9.97E+36	9.97E+36	-2.147E+09
2018	51	16	18	51	51	-86.706512	-101.98278	0	9.97E+36	9.97E+36	9.97E+36	9.97E+36	9.97E+36	-2.147E+09
2018	51	16	18	51	51	-86.729759	-100.04027	0	9.97E+36	9.97E+36	9.97E+36	9.97E+36	9.97E+36	-2.147E+09
2018	51	16	18	51	51	-86.748924	-98.108452	0	9.97E+36	9.97E+36	9.97E+36	9.97E+36	9.97E+36	-2.147E+09
2018	51	16	18	51	51	-86.764153	-96.1912	0	9.97E+36	9.97E+36	9.97E+36	9.97E+36	9.97E+36	-2.147E+09
2018	51	16	18	51	51	-86.775604	-94.292236	0	9.97E+36	9.97E+36	9.97E+36	9.97E+36	9.97E+36	-2.147E+09
2018	51	16	18	51	51	-86.78344	-92.415031	0	9.97E+36	9.97E+36	9.97E+36	9.97E+36	9.97E+36	-2.147E+09
2018	51	16	18	51	51	-86.787804	-90.562828	0	9.97E+36	9.97E+36	9.97E+36	9.97E+36	9.97E+36	-2.147E+09
2018	51	16	18	51	51	-86.788864	-88.738579	0	9.97E+36	9.97E+36	9.97E+36	9.97E+36	9.97E+36	-2.147E+09
2018	51	16	18	51	51	-86.786789	-86.944946	0	9.97E+36	9.97E+36	9.97E+36	9.97E+36	9.97E+36	-2.147E+09
2018	51	16	18	51	51	-86.781723	-85.18478	0	9.97E+36	9.97E+36	9.97E+36	9.97E+36	9.97E+36	-2.147E+09

Este código guarda un archivo.csv, el cual se puede abrir con Excel, algún editor de texto, u otros códigos o software

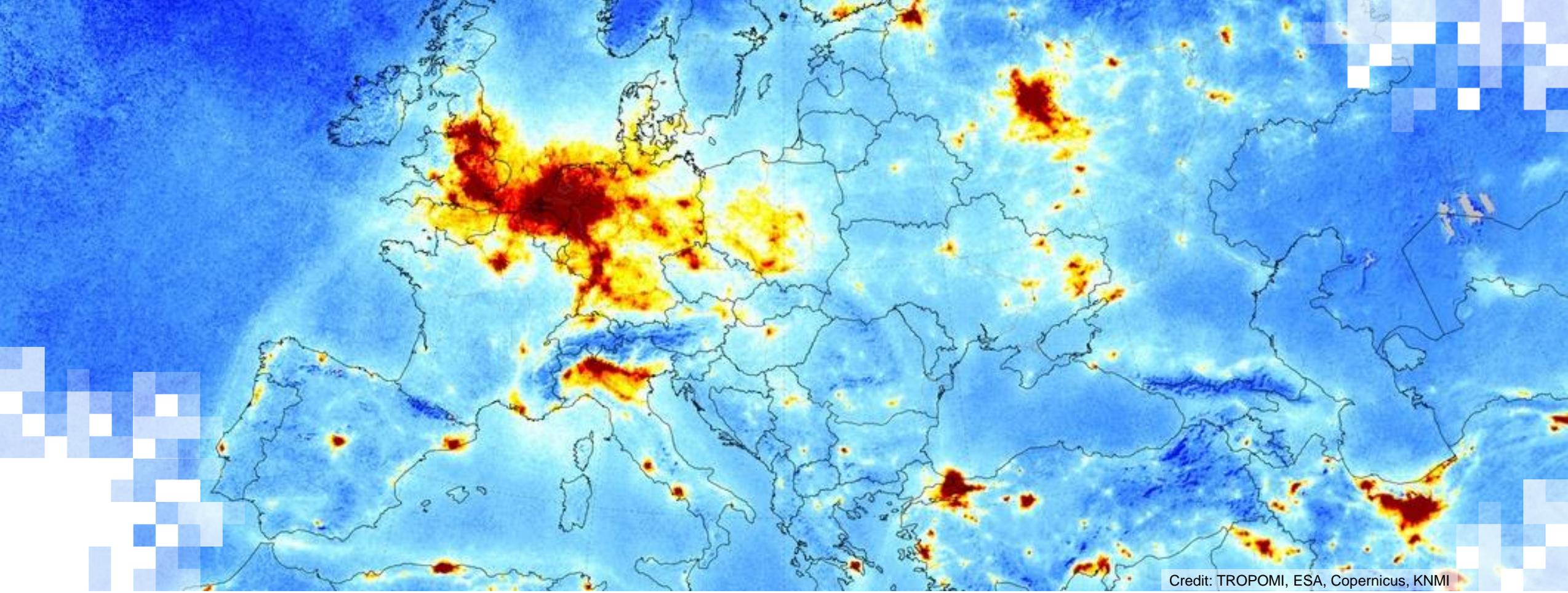


Editando el Código

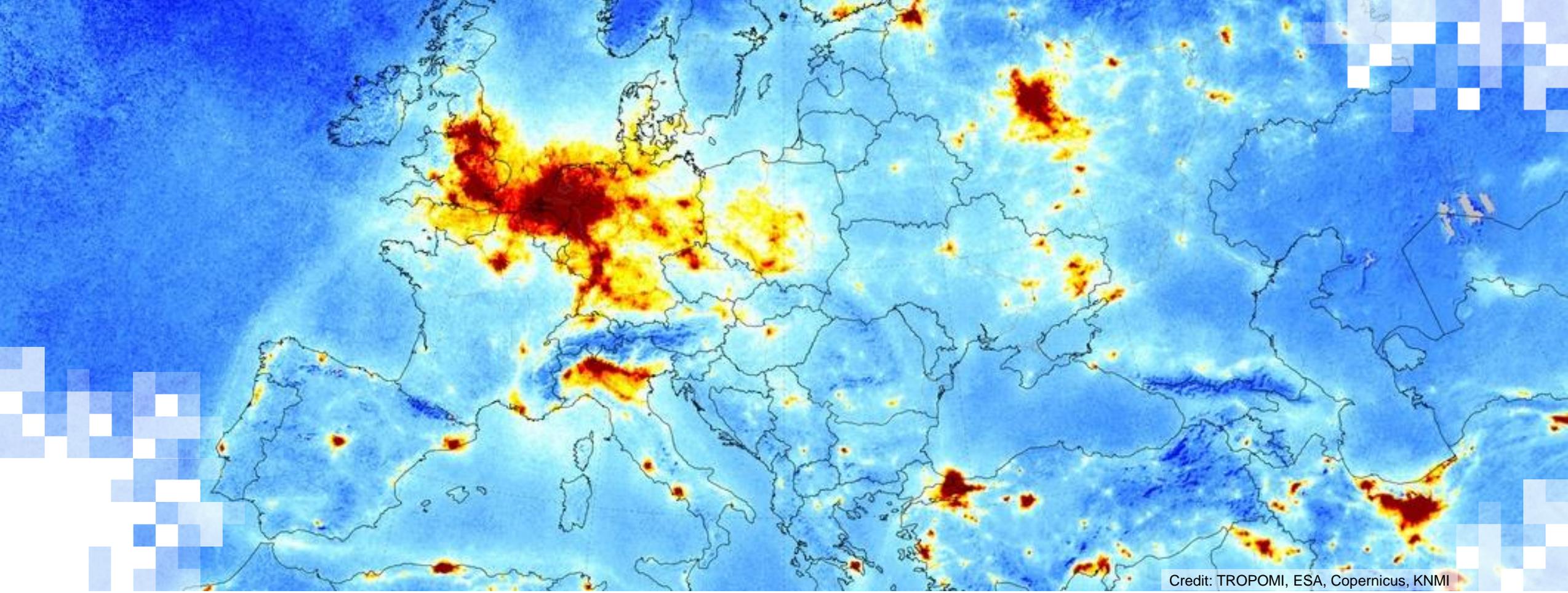
Cambie el SDS a ser escrito como salida

NOTE: Este código funcionará únicamente cuando las variable listadas sean la misma dimensión. Use el código “list SDS” para ver las dimensiones de las variables

```
27
28 #loops through all files listed in the text file
29 for FILE_NAME in fileList:
30     FILE_NAME=FILE_NAME.strip()
31     user_input=input('\nWould you like to process\n' + FILE_NAME + '\n\n(Y/N)')
32     if(user_input == 'N' or user_input == 'n'):
33         print('Skipping...')
34         continue
35     else:
36         file = Dataset(FILE_NAME, 'r')
37     # read the data
38     if 'NO2' in FILE_NAME:
39         print('This is an TROPOMI NO2 file.')
40         #this is how you access the data tree in an hdf5 file
41         SDS_NAME='nitrogendioxide_tropospheric_column'
42     elif 'AER_AI' in FILE_NAME:
43         print('This is an TROPOMI Aerosol Index file.')
44         SDS_NAME='aerosol_index_354_388'
45     ds=file
46     grp='PRODUCT'
47     lat= ds.groups[grp].variables['latitude'][0][:][:]
48     lon= ds.groups[grp].variables['longitude'][0][:][:]
49     data= ds.groups[grp].variables[SDS_NAME]
50
51     #get necessary attributes
52     fv=data._FillValue
53
```



Transición a Datos OMI



Leer un Archivo OMI de NO₂ (he5) e Imprimir
Lista de SDS

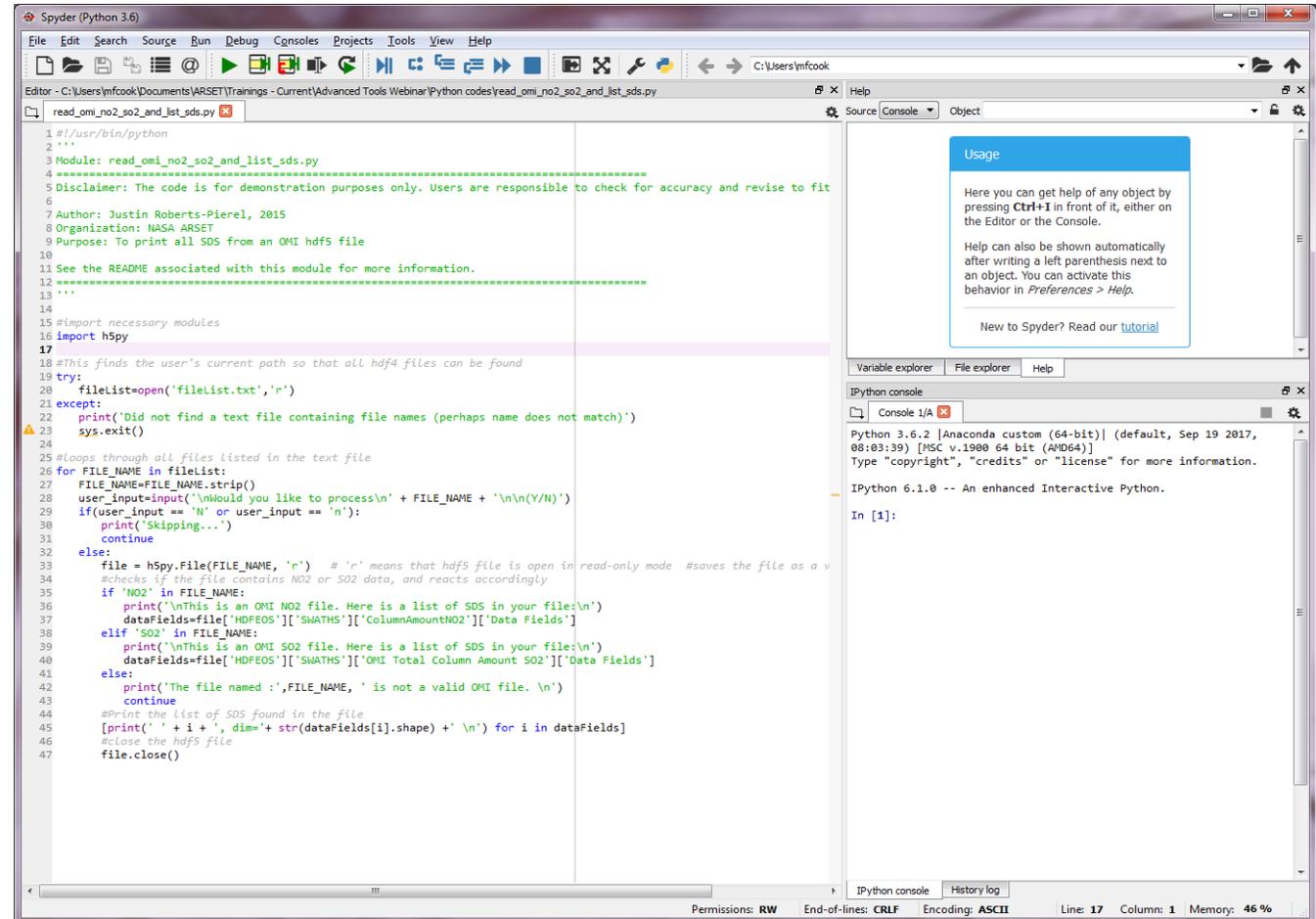
Imprimir Conjuntos de Datos Científicos (Scientific Data Sets o SDSs)

read_omi_no2_so2_and_list_sds.py
read_omi_no2_so2_and_list_sds_geo.py

Propósito: leer archivos de datos OMI de NO₂ o SO₂ nivel 2 en formato hdf e imprimir todos los **Conjuntos de Datos Científicos (Scientific Data Sets o SDS)**.

En su forma actual, todos estos códigos sirven *solo para productos nivel 2, no productos en cuadrícula.*

El código ‘_geo.py’ lista todos los campos de geolocalización



```
1 #!/usr/bin/python
2 ...
3 Module: read_omi_no2_so2_and_list_sds.py
4 -----
5 Disclaimer: The code is for demonstration purposes only. Users are responsible to check for accuracy and revise to fit
6
7 Author: Justin Roberts-Pierel, 2015
8 Organization: NASA ARSET
9 Purpose: To print all SDS from an OMI hdf5 file
10
11 See the README associated with this module for more information.
12 -----
13 ...
14
15 #import necessary modules
16 import h5py
17
18 #This finds the user's current path so that all hdf4 files can be found
19 try:
20     fileList=open('fileList.txt','r')
21 except:
22     print('Did not find a text file containing file names (perhaps name does not match)')
23     sys.exit()
24
25 #Loops through all files listed in the text file
26 for FILE_NAME in fileList:
27     FILE_NAME=FILE_NAME.strip()
28     user_input=input('\nWould you like to process\n' + FILE_NAME + '\n\n(Y/N)')
29     if user_input == 'N' or user_input == 'n':
30         print('Skipping...')
31         continue
32     else:
33         file = h5py.File(FILE_NAME, 'r') # 'r' means that hdf5 file is open in read-only mode #saves the file as a v
34         #checks if the file contains NO2 or SO2 data, and reacts accordingly
35         if 'NO2' in FILE_NAME:
36             print('\nThis is an OMI NO2 file. Here is a list of SDS in your file:\n')
37             dataFields=file['HDFEOS']['SWATHS']['ColumnAmountNO2']['Data Fields']
38         elif 'SO2' in FILE_NAME:
39             print('\nThis is an OMI SO2 file. Here is a list of SDS in your file:\n')
40             dataFields=file['HDFEOS']['SWATHS']['OMI Total Column Amount SO2']['Data Fields']
41         else:
42             print('The file named :,FILE_NAME, ' is not a valid OMI file. \n')
43             continue
44         #Print the list of SDS found in the file
45         [print(' ' + i + ', dim='+ str(dataFields[i].shape) + ' \n') for i in dataFields]
46         #close the hdf5 file
47         file.close()
```



Ejecución y Salida

- Haga clic en la flecha verde para ejecutar el código
- El código procesará todos los archivos en `fileList.txt` uno por uno
- Siga las instrucciones en la terminal `ipython` (o sea, teclee 'Y' o 'N' cuando se le pida y presione Enter)

The screenshot displays the Spyder Python IDE interface. The top menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Projects, Tools, View, and Help. The Run menu is open, and the green play button (Run) is highlighted with a red box. A green arrow points from the first bullet point of the text to this button. The editor window shows a Python script with the following content:

```
1 #!/usr/bin/python
2 ...
3 Module: read_omi_no2_so2_and_list_sds.py
4 -----
5 Disclaimer: The code is for demonstration purposes only. Users are responsible to check for accuracy and revise to fit their objective.
6
7 Author: Justin Roberts-Pierel, 2015
8 Organization: NASA ARSET
9 Purpose: To print all SDS from an OMI hdf5 file
10
11 See the README associated with this module for more information.
12 -----
13 ...
14
15 #import necessary modules
16 import h5py
17
18 #This finds the user's current path so that all hdf4 files can be found
19 try:
20     fileList=open('fileList.txt','r')
21 except:
22     print('Did not find a text file containing file names (perhaps name does not match)')
23     sys.exit()
24
25 #Loops through all files listed in the text file
26 for FILE_NAME in fileList:
27     FILE_NAME=FILE_NAME.strip()
28     user_input=input('\nWould you like to process\n' + FILE_NAME + '\n\n(Y/N)')
29     if(user_input == 'N' or user_input == 'n'):
30         print('Skipping...')
31         continue
32     else:
33         file = h5py.File(FILE_NAME, 'r') # 'r' means that hdf5 file is open in read-only mode #saves the file as a variable named 'hdf'
34         #checks if the file contains NO2 or SO2 data, and reacts accordingly
35         if 'NO2' in FILE_NAME:
36             print('\nThis is an OMI NO2 file. Here is a list of SDS in your file:\n')
37             dataFields=file['HDFEOS']['SWATHS']['ColumnAmountNO2']['Data Fields']
38         elif 'SO2' in FILE_NAME:
39             print('\nThis is an OMI SO2 file. Here is a list of SDS in your file:\n')
40             dataFields=file['HDFEOS']['SWATHS']['OMI Total Column Amount SO2']['Data Fields']
41         else:
42             print('The file named:',FILE_NAME, ' is not a valid OMI file. \n')
43             continue
44         #Print the List of SDS found in the file
45         [print(' ' + i + ', dim='+ str(dataFields[i].shape) + ' \n') for i in dataFields]
46         #close the hdf5 file
47         file.close()
```

The IPython console on the right shows the output of the script, listing various data fields and their dimensions. The output is highlighted with a red box and labeled "Salida":

```
SlantColumnAmountNO2Std, dim=(1644, 60)
SmallPixelRadiance, dim=(10, 60)
SmallPixelRadiancePointer, dim=(1644, 2)
TerrainHeight, dim=(1644, 60)
TerrainPressure, dim=(1644, 60)
TerrainReflectivity, dim=(1644, 60)
TropopausePressure, dim=(1644, 60)
VcdApBelowCloud, dim=(1644, 60)
VcdApStrat, dim=(1644, 60)
VcdApTrop, dim=(1644, 60)
VcdQualityFlags, dim=(1644, 60)
WavelengthRegistrationCheck, dim=(1644, 60)
WavelengthRegistrationCheckStd, dim=(1644, 60)
XTrackQualityFlags, dim=(1644, 60)

Would you like to process
OMI-Aura_L2-OMNO2_2015m0615t2010-o58069_v003-2016m0821t121351.he5
(Y/N)
```

The status bar at the bottom indicates: Permissions: RW End-of-lines: CRLF Encoding: ASCII Line: 17 Column: 1 Memory: 46 %

Editando el Código

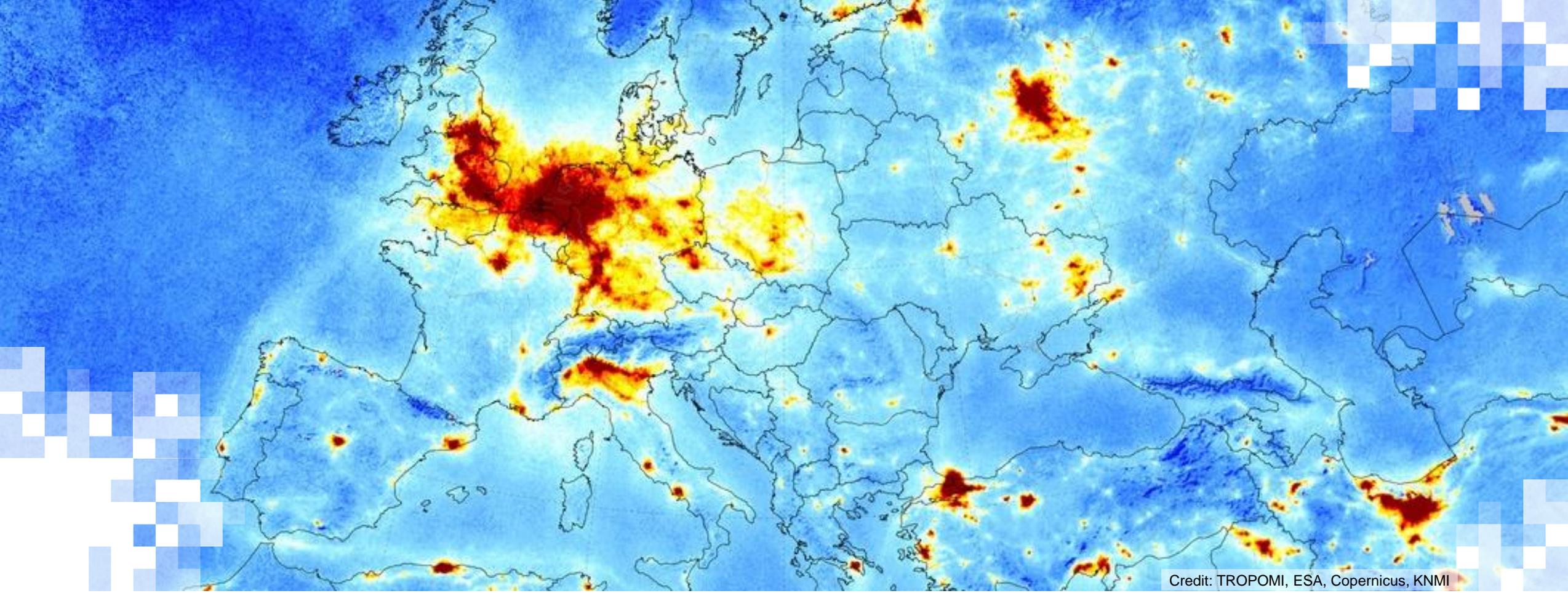
Cambie el nombre de fileList.txt a lo que usted quiera

```
1 #!/usr/bin/python
2 '''
3 Module: read_omi_no2_so2_and_list_sds.py
4 -----
5 Disclaimer: The code is for demonstration purposes only. Users are responsible to check for accuracy and revise to fit their objective.
6
7 Author: Justin Roberts-Pierel, 2015
8 Organization: NASA ARSET
9 Purpose: To print all SDS from an OMI hdf5 file
10
11 See the README associated with this module for more information.
12 -----
13 '''
14
15 #import necessary modules
16 import h5py
17
18 #This finds the user's current path so that all hdf5 files can be found
19 try:
20     fileList=open('fileList.txt','r')
21 except:
22     print('Did not find a text file containing file names (perhaps name does not match)')
23     sys.exit()
24
25 #Loops through all files listed in the text file
26 for FILE_NAME in fileList:
27     FILE_NAME=FILE_NAME.strip()
28     user_input=input('\nWould you like to process\n' + FILE_NAME + '\n\n(Y/N)')
29     if(user_input == 'N' or user_input == 'n'):
30         print('Skipping...')
31         continue
32     else:
33         file = h5py.File(FILE_NAME, 'r') # 'r' means that hdf5 file is open in read-only mode #saves the file as a variable named 'hdf'
34         #checks if the file contains NO2 or SO2 data, and reacts accordingly
35         if 'NO2' in FILE_NAME:
36             print('\nThis is an OMI NO2 file. Here is a list of SDS in your file:\n')
37             dataFields=file['HDFEOS']['SWATHS']['ColumnAmountNO2']['Data Fields']
38         elif 'SO2' in FILE_NAME:
39             print('\nThis is an OMI SO2 file. Here is a list of SDS in your file:\n')
40             dataFields=file['HDFEOS']['SWATHS']['OMI Total Column Amount SO2']['Data Fields']
41         else:
42             print('The file named :',FILE_NAME, ' is not a valid OMI file. \n')
43             continue
44         #Print the list of SDS found in the file
45         [print(' ' + i + ' ', dim=' ' + str(dataFields[i].shape) + ' \n') for i in dataFields]
46         #close the hdf5 file
47         file.close()
```

Al cambiar la ubicación de los dataFields a geolocalización (se puede encontrar en otros códigos) esto también puede listar las variables de geolocalización disponibles

Aplicaciones

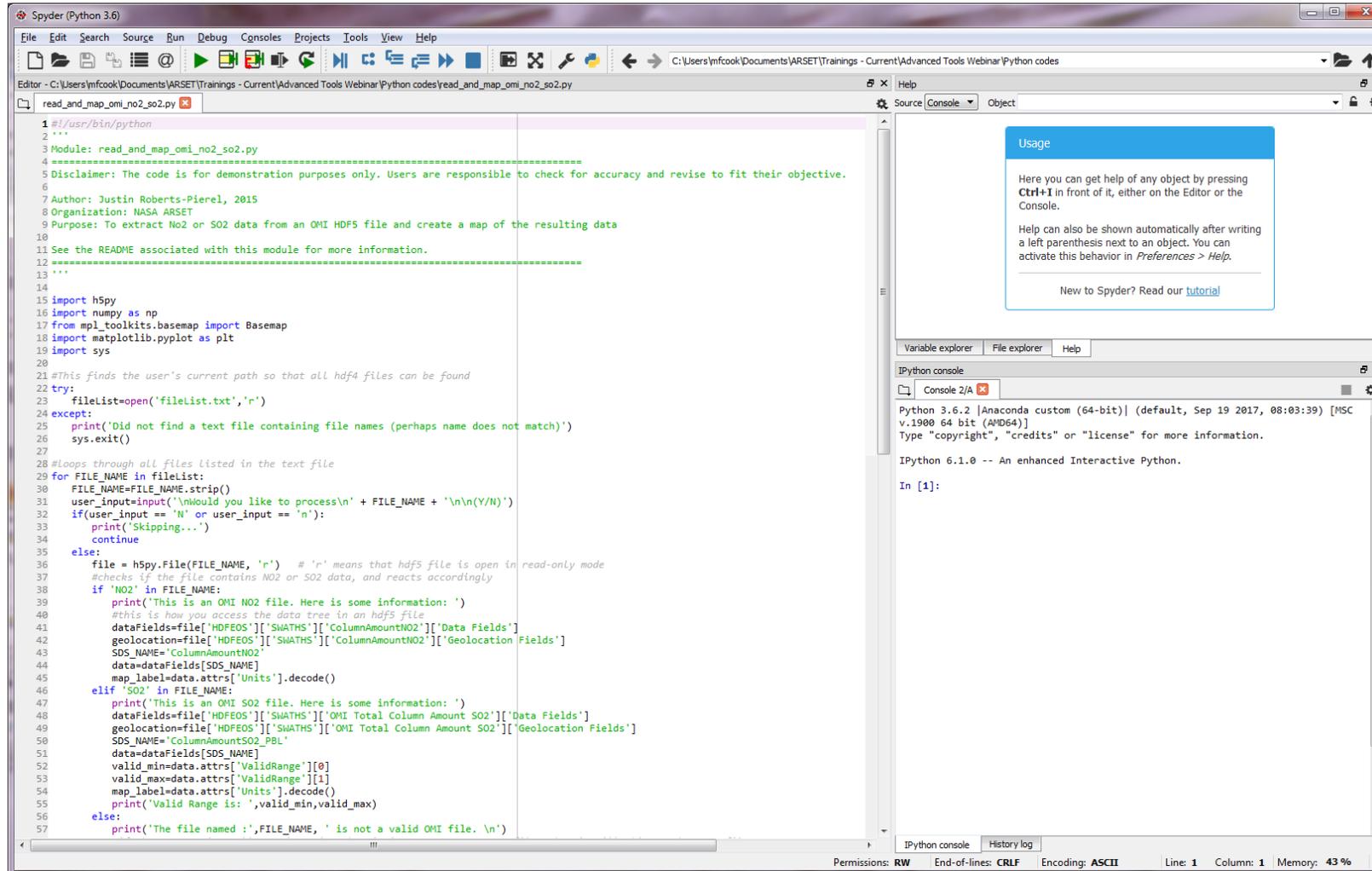
- Los datos OMI Nivel 2 de NO₂ y SO₂ se presentan en archivos de formato hdf
- Cada archivo HDF contiene varios parámetros geofísicos
- Se necesitan códigos/herramientas especiales para abrir los archivos hdf
- Este código ayuda a los usuarios a ver los nombres y dimensiones de los SDSs disponibles dentro de un archivo hdf para análisis adicional



Mapear NO_2 o SO_2

Diagramar y Guardar un Mapa de OMI NO₂ o SO₂

read_and_map_omi_so2_no2.py



```
1 #!/usr/bin/python
2 ...
3 Module: read_and_map_omi_no2_so2.py
4 =====
5 Disclaimer: The code is for demonstration purposes only. Users are responsible to check for accuracy and revise to fit their objective.
6
7 Author: Justin Roberts-Pierel, 2015
8 Organization: NASA ARSET
9 Purpose: To extract No2 or SO2 data from an OMI HDF5 file and create a map of the resulting data
10
11 See the README associated with this module for more information.
12 =====
13 ...
14 ...
15 import h5py
16 import numpy as np
17 from mpl_toolkits.basemap import Basemap
18 import matplotlib.pyplot as plt
19 import sys
20
21 #This finds the user's current path so that all hdf4 files can be found
22 try:
23     fileList=open('fileList.txt','r')
24 except:
25     print('Did not find a text file containing file names (perhaps name does not match)')
26     sys.exit()
27
28 #Loops through all files listed in the text file
29 for FILE_NAME in fileList:
30     FILE_NAME=FILE_NAME.strip()
31     user_input=input('\nWould you like to process\n' + FILE_NAME + '\n\n(Y/N)')
32     if(user_input == 'N' or user_input == 'n'):
33         print('Skipping...')
34         continue
35     else:
36         file = h5py.File(FILE_NAME, 'r') # 'r' means that hdf5 file is open in read-only mode
37         #checks if the file contains NO2 or SO2 data, and reacts accordingly
38         if 'NO2' in FILE_NAME:
39             print('This is an OMI NO2 file. Here is some information: ')
40             #this is how you access the data tree in an hdf5 file
41             dataFields=file['HDFEOS']['SWATHS']['ColumnAmountNO2']['Data Fields']
42             geoLocation=file['HDFEOS']['SWATHS']['ColumnAmountNO2']['GeoLocation Fields']
43             SDS_NAME='ColumnAmountNO2'
44             data=dataFields[SDS_NAME]
45             map_label=data.attrs['Units'].decode()
46         elif 'SO2' in FILE_NAME:
47             print('This is an OMI SO2 file. Here is some information: ')
48             dataFields=file['HDFEOS']['SWATHS']['OMI Total Column Amount SO2']['Data Fields']
49             geoLocation=file['HDFEOS']['SWATHS']['OMI Total Column Amount SO2']['GeoLocation Fields']
50             SDS_NAME='ColumnAmountSO2_PBL'
51             data=dataFields[SDS_NAME]
52             valid_min=data.attrs['ValidRange'][0]
53             valid_max=data.attrs['ValidRange'][1]
54             map_label=data.attrs['Units'].decode()
55             print('Valid Range is: ',valid_min,valid_max)
56         else:
57             print('The file named:',FILE_NAME, ' is not a valid OMI file. \n')
```

Ejecución y Salida

```
In [1]: runfile('C:/Users/mfcook/Documents/ARSET/Trainings - Current/Advanced Tools Webinar/Python codes/  
read_and_map_omi_no2_so2.py', wdir='C:/Users/mfcook/Documents/ARSET/Trainings - Current/Advanced Tools Webinar/  
Python codes')
```

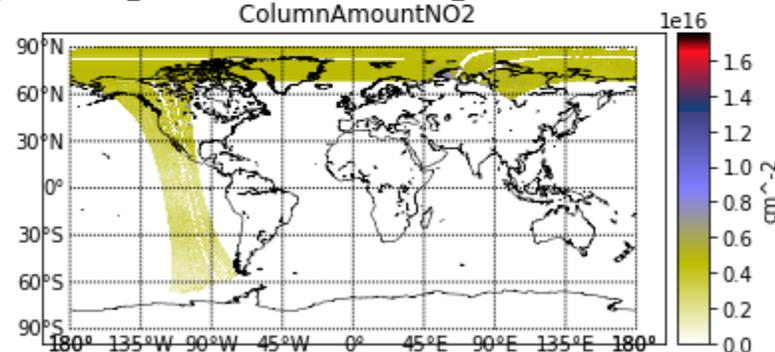
```
Would you like to process  
OMI-Aura_L2-OMNO2_2008m0720t2016-o21357_v003-2016m0820t102252.he5
```

```
(Y/N)Y
```

```
This is an OMI NO2 file. Here is some information:  
3.14792e+15  
The average of this data is: 3.14792e+15  
The standard deviation is: 1.35182e+15  
The median is: 2.90004e+15  
The range of latitude in this file is: -75.0061 to 89.8693 degrees  
The range of longitude in this file is: -179.99 to 179.975 degrees
```

```
Would you like to create a map of this data? Please enter Y or N  
Y
```

```
OMI-Aura_L2-OMNO2_2008m0720t2016-o21357_v003-2016m0820t102252.he5  
ColumnAmountNO2
```



```
Would you like to save this map? Please enter Y or N  
|
```

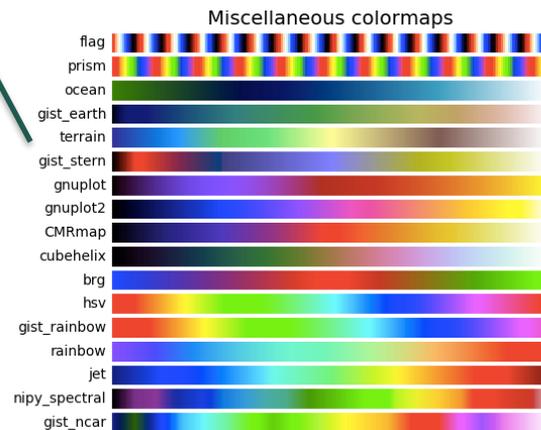
NO₂/SO₂- estadísticas

Mapa de salida

Editando el Código

Cambie la escala cromática

```
93 if is_map == 'Y' or is_map == 'y':
94     data = np.ma.masked_array(data, np.isnan(data))
95     m = Basemap(projection='cyl', resolution='l',
96                 llcrnrlat=-90, urcrnrlat = 90,
97                 llcrnrlon=-180, urcrnrlon = 180)
98     m.drawcoastlines(linewidth=0.5)
99     m.drawparallels(np.arange(-90., 120., 30.), labels=[1, 0, 0, 0])
100    m.drawmeridians(np.arange(-180, 180., 45.), labels=[0, 0, 0, 1])
101    my_cmap = plt.cm.get_cmap('gist_stern_r')
102    my_cmap.set_under('w')
103    m.pcolormesh(lon, lat, data, latlon=True, vmin=0, vmax=np.nanmax(dat
104    cb = m.colorbar()
105    cb.set_label(map_label)
106    plt.autoscale()
107    #title the plot
108    plt.title('{0}\n {1}'.format(FILE_NAME,
109    fig = plt.gcf()
110    # Show the plot window.
111    plt.show()
112    #once you close the map it asks if you'd l
113    is_save=str(input('\nWould you like to
114    if is_save == 'Y' or is_save == 'y':
115        #saves as a png if the user would l
116        pngfile = '{0}.png'.format(FILE_NAM
117        fig.savefig(pngfile)
118    #close the hdf5 file
119    file.close()
```



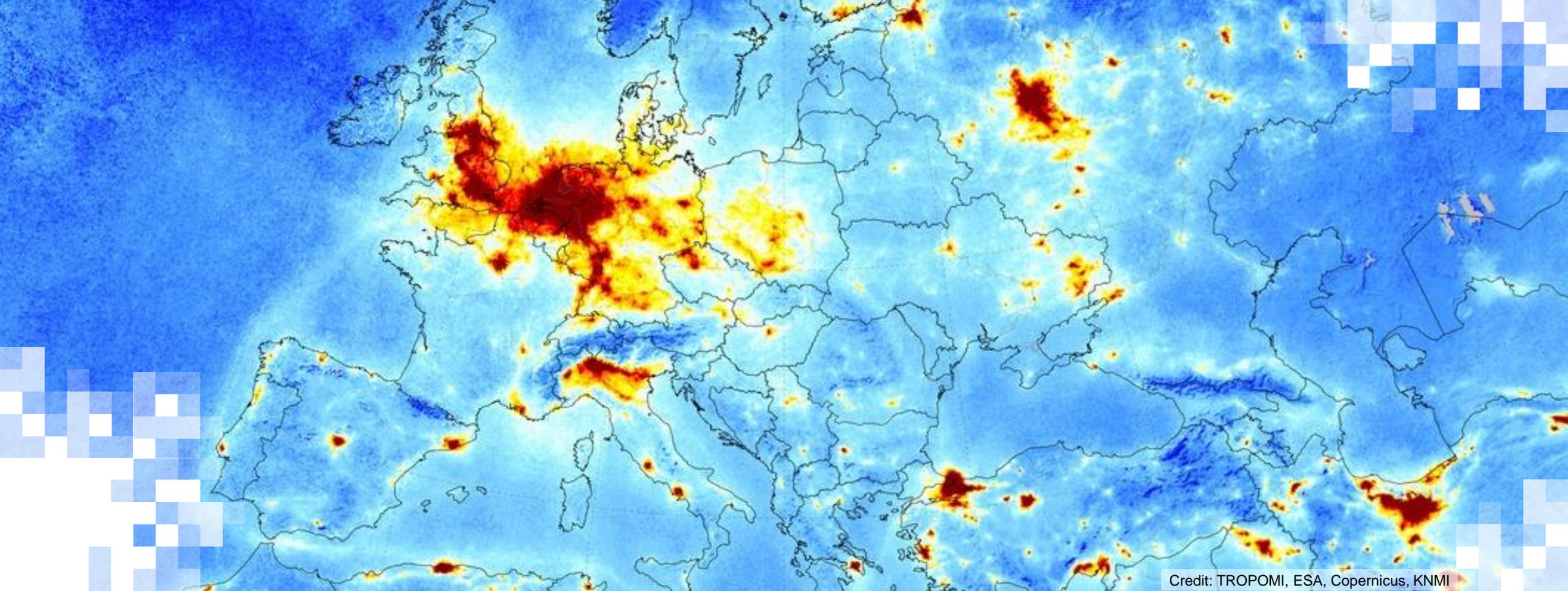
Cambie el SDS a diagramar

```
28 #Loops through all files listed in the text file
29 for FILE_NAME in fileList:
30     FILE_NAME=FILE_NAME.strip()
31     user_input=input('\nWould you like to process\n' + FILE_NAME + '\n\n(Y/N)')
32     if(user_input == 'N' or user_input == 'n'):
33         print('Skipping...')
34         continue
35     else:
36         file = h5py.File(FILE_NAME, 'r') # 'r' means that hdf5 file is open in read-only mode
37         #checks if the file contains NO2 or SO2 data, and reacts accordingly
38         if 'NO2' in FILE_NAME:
39             print('This is an OMI NO2 file. Here is some information: ')
40             #this is how you access the data tree in an hdf5 file
41             dataFields=file['HDFEOS']['SWATHS']['ColumnAmountNO2']['Data Fields']
42             geolocation=file['HDFEOS']['SWATHS']['ColumnAmountNO2']['Geolocation Fields']
43             SDS_NAME='ColumnAmountNO2'
44             data=dataFields[SDS_NAME]
45             map_label=data.attrs['Units'].decode()
46         elif 'SO2' in FILE_NAME:
47             print('This is an OMI SO2 file. Here is some information: ')
48             dataFields=file['HDFEOS']['SWATHS']['OMI Total Column Amount SO2']['Data Fields']
49             geolocation=file['HDFEOS']['SWATHS']['OMI Total Column Amount SO2']['Geolocation Fields']
50             SDS_NAME='ColumnAmountSO2_PBL'
51             data=dataFields[SDS_NAME]
52             valid_min=data.attrs['ValidRange'][0]
53             valid_max=data.attrs['ValidRange'][1]
54             map_label=data.attrs['Units'].decode()
55             print('Valid Range is: ',valid_min,valid_max)
56         else:
57             print('The file named: ',FILE_NAME, ' is not a valid OMI file. \n')
58             #if the program is unable to determine that it is an OMI SO2 or NO2 file, then it will skip
59             continue
```

https://matplotlib.org/examples/color/colormaps_reference.html

Aplicaciones

- Este es un ejemplo de código para leer y mapear datos de NO₂ y SO₂ Nivel 2 de TROPOMI
- Se puede modificar el código para diferentes necesidades cartográficas
- El usuario puede crear mapas diarios de columnas de gases trazadores sobre ciertas regiones y empezar a analizar cambios a través del tiempo
- Estos mapas también pueden ayudar a identificar regiones de alta contaminación



Extraer NO_2/SO_2 en una Ubicación Particular

Extraer Valores del AOD* de Datos de Aerosoles Nivel 2 de MODIS

read_mod_aerosol_and_list_sds.py

- **Propósito:** leer un archivo de datos OMI NO₂/SO₂ nivel 2 en formato HDF y extraer valores para una ubicación particular en el suelo

*Siglas de Aerosol Optical Depth, Espesor óptico de aerosoles, en inglés

```
1 #!/usr/bin/python
2 ...
3 Module: read_omi_no2_so2_at_a_location.py
4 -----
5 Disclaimer: The code is for demonstration purposes only. Users are responsible to check for accuracy and revise to
6
7 Author: Justin Roberts-Pierel, 2015
8 Organization: NASA ARSET
9 Purpose: To view info about a variety of SDS from an OMI hdf file both generally and at a specific lat/lon
10
11 See the README associated with this module for more information.
12 -----
13 ...
14
15 #import necessary modules
16 import h5py
17 import numpy as np
18 import sys
19 from numpy import unravel_index
20
21 #This finds the user's current path so that all hdf4 files can be found
22 try:
23     filelist=open('filelist.txt','r')
24 except:
25     print('Did not find a text file containing file names (perhaps name does not match)')
26     sys.exit()
27
28 #loops through all files listed in the text file
29 for FILE_NAME in filelist:
30     FILE_NAME=FILE_NAME.strip()
31     user_input=input('\nWould you like to process\n' + FILE_NAME + '\n\n(Y/N)')
32     if user_input == 'N' or user_input == 'n':
33         print('Skipping...')
34         continue
35     else:
36         file = h5py.File(FILE_NAME, 'r') # 'r' means that hdf5 file is open in read-only mode
37         if 'NO2' in FILE_NAME:
38             print('This is an OMI NO2 file. Here is some information: ')
39             #this is how you access the data tree in an hdf5 file
40             dataFields=file['HDFEOS']['SWATHS']['ColumnAmountNO2']['Data Fields']
41             geolocation=file['HDFEOS']['SWATHS']['ColumnAmountNO2']['Geolocation Fields']
42             SDS_NAME='ColumnAmountNO2'
43             data=dataFields[SDS_NAME]
44             map_label=data.attrs['Units'].decode()
45         elif 'SO2' in FILE_NAME:
46             print('This is an OMI SO2 file. Here is some information: ')
47             dataFields=file['HDFEOS']['SWATHS']['OMI Total Column Amount SO2']['Data Fields']
48             geolocation=file['HDFEOS']['SWATHS']['OMI Total Column Amount SO2']['Geolocation Fields']
49             SDS_NAME='ColumnAmountSO2_PBL'
50             data=dataFields[SDS_NAME]
51             valid_min=data.attrs['ValidRange'][0]
52             valid_max=data.attrs['ValidRange'][1]
53             map_label=data.attrs['Units'].decode()
54             print('Valid Range is: ',valid_min,valid_max)
55         else:
56             print('The file named ',FILE_NAME, ' is not a valid OMI file. \n')
57             #if the program is unable to determine that it is an OMI SO2 or NO2 file, then it will skip to the next
```

```
Python 3.6.2 [Anaconda custom (64-bit)] (default, Sep 19 2017, 08:03:39) [MSC v.1900 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 6.1.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/mfcook/Documents/ARSET/Trainings - Current/Advanced Tools Webinar/Python codes/
read_omi_no2_so2_at_a_location.py', wdir='C:/Users/mfcook/Documents/ARSET/Trainings - Current/Advanced Tools Webinar/Python
codes')

Would you like to process
OMI-Aura_L2-OMNO2_2008m0720t2016-021357_v003-2016m0820t102252.he5

(Y/N)Y
This is an OMI NO2 file. Here is some information:
The range of latitude in this file is: -75.0861 to 89.8693 degrees
The range of longitude in this file is: -179.99 to 179.975 degrees

Please enter the latitude you would like to analyze (Deg. N): 30

Please enter the longitude you would like to analyze (Deg. E): -100
855 59

The nearest pixel to your entered location is at:
Latitude: 29.8233 Longitude: -101.774
The value of ColumnAmountNO2 at this pixel is 3.92950208633e+15
There are 9 valid pixels in a 3x3 grid centered at your entered location.
The average value in this grid is: 4.15249517773e+15
The median value in this grid is: 4.01630659961e+15
The standard deviation in this grid is: 2.77808737236e+14

There are 25 valid pixels in a 5x5 grid centered at your entered location.
The average value in this grid is: 4.054780255004e+15
The median value in this grid is: 3.96426125666e+15
The standard deviation in this grid is: 4.46095029635e+14

Would you like to process
OMI-Aura_L2-OMNO2_2015m0615t2010-058069_v003-2016m0821t121351.he5

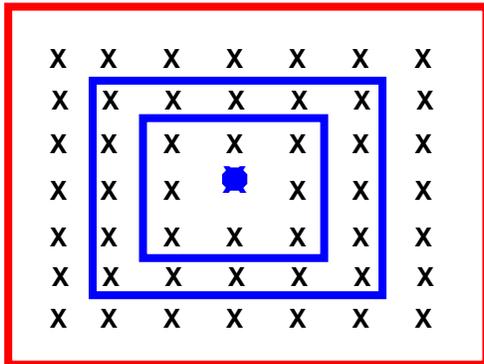
(Y/N)Y
```

Ejecución y Salida

Teclee “Y” para procesar el archivo, “N” para saltar

Lat. y Lon. de la estación

Salidas



```
Would you like to process
OMI-Aura_L2-OMNO2_2008m0720t2016-o21357_v003-2016m0820t102252.he5

(Y/N)Y
This is an OMI NO2 file. Here is some information:
The range of latitude in this file is: -75.0061 to 89.8693 degrees
The range of longitude in this file is: -179.99 to 179.975 degrees
```

```
Please enter the latitude you would like to analyze (Deg. N): 30
Please enter the longitude you would like to analyze (Deg. E): -100
855 59
```

```
The nearest pixel to your entered location is at:
Latitude: 29.8233 Longitude: -101.774
The value of ColumnAmountNO2 at this pixel is 3.92950208633e+15
There are 9 valid pixels in a 3x3 grid centered at your entered location.
The average value in this grid is: 4.15249517773e+15
The median value in this grid is: 4.01630659661e+15
The standard deviation in this grid is: 2.77808737236e+14
```

```
There are 25 valid pixels in a 5x5 grid centered at your entered location.

The average value in this grid is: 4.05478825804e+15
The median value in this grid is: 3.96426125666e+15
The standard deviation in this grid is: 4.40095029635e+14
```

```
Would you like to process
OMI-Aura_L2-OMNO2_2015m0615t2010-o58069_v003-2016m0821t121351.he5
```

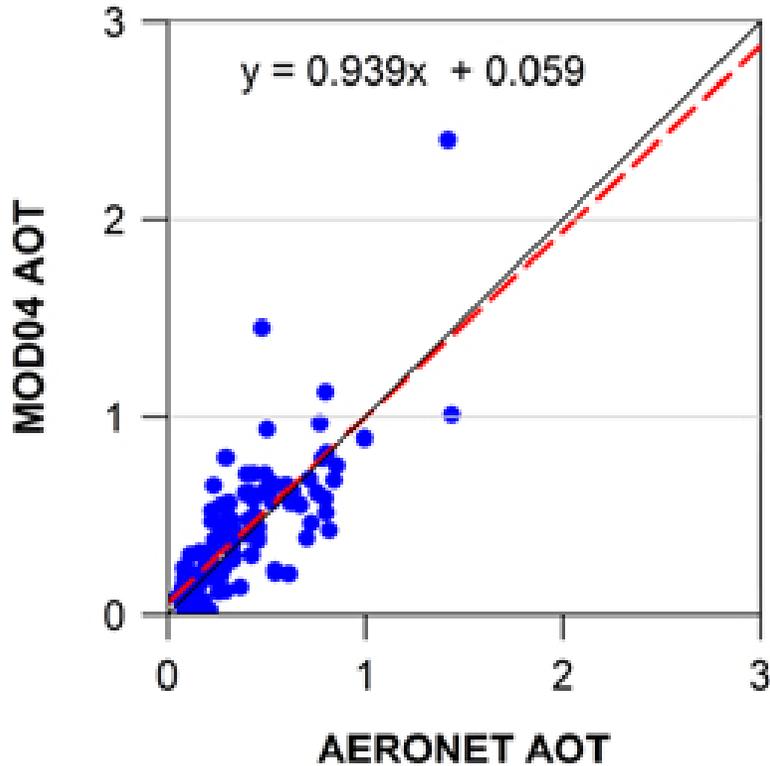
```
(Y/N)
```

Editando el Código – Cambiar de SDS

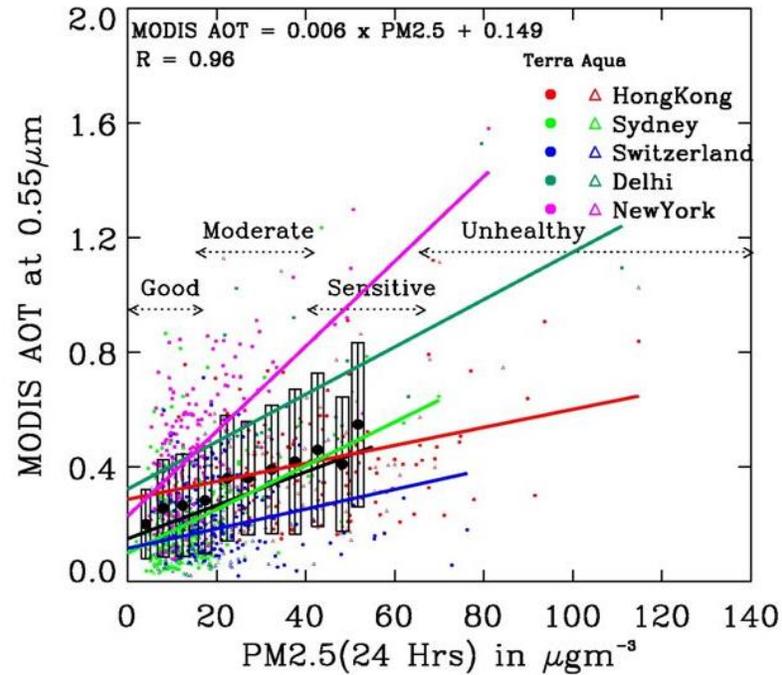
```
27
28 #Loops through all files listed in the text file
29 for FILE_NAME in fileList:
30     FILE_NAME=FILE_NAME.strip()
31     user_input=input('\nWould you like to process\n' + FILE_NAME + '\n\n(Y/N)')
32     if(user_input == 'N' or user_input == 'n'):
33         print('Skipping...')
34         continue
35     else:
36         file = h5py.File(FILE_NAME, 'r') # 'r' means that hdf5 file is open in read-only mode
37         if 'NO2' in FILE_NAME:
38             print('This is an OMI NO2 file. Here is some information: ')
39             #this is how you access the data tree in an hdf5 file
40             dataFields=file['HDFEOS']['SWATHS']['ColumnAmountNO2']['Data Fields']
41             geolocation=file['HDFEOS']['SWATHS']['ColumnAmountNO2']['Geolocation Fields']
42             SDS_NAME='ColumnAmountNO2'
43             data=dataFields[SDS_NAME]
44             map_label=data.attrs['Units'].decode()
45         elif 'SO2' in FILE_NAME:
46             print('This is an OMI SO2 file. Here is some information: ')
47             dataFields=file['HDFEOS']['SWATHS']['OMI Total Column Amount SO2']['Data Fields']
48             geolocation=file['HDFEOS']['SWATHS']['OMI Total Column Amount SO2']['Geolocation Fields']
49             SDS_NAME='ColumnAmountSO2_PBL'
50             data=dataFields[SDS_NAME]
51             valid_min=data.attrs['ValidRange'][0]
52             valid_max=data.attrs['ValidRange'][1]
53             map_label=data.attrs['Units'].decode()
54             print('Valid Range is: ',valid_min,valid_max)
55         else:
56             print('The file named :',FILE_NAME, ' is not a valid OMI file. \n')
57             #if the program is unable to determine that it is an OMI SO2 or NO2 file, then it will skip to the next file
58             continue
```

Aplicaciones

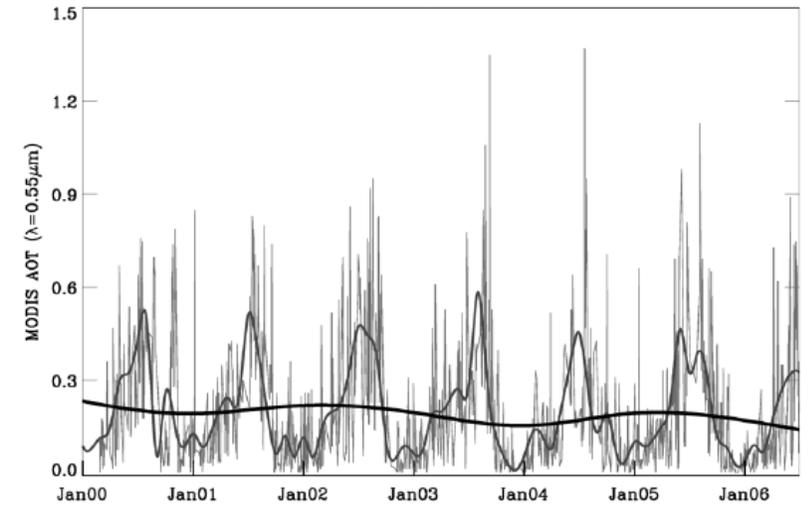
Validación Satelital del AOD



Relación AOD-PM_{2.5}

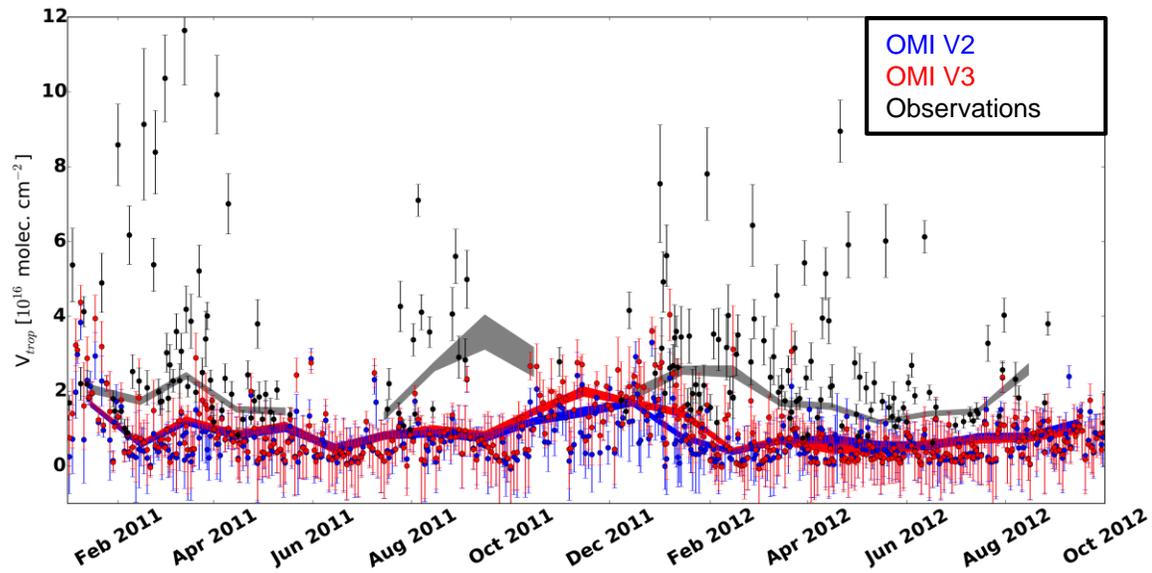


Análisis de Series Temporales



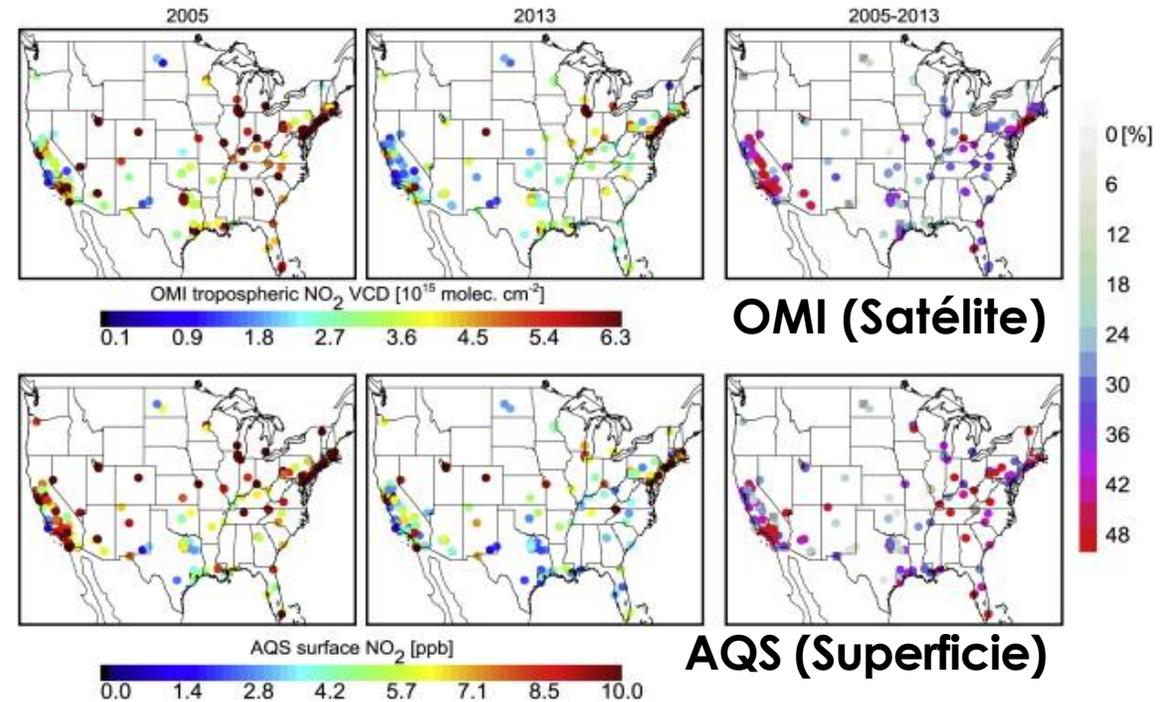
Aplicaciones

Validación Satelital

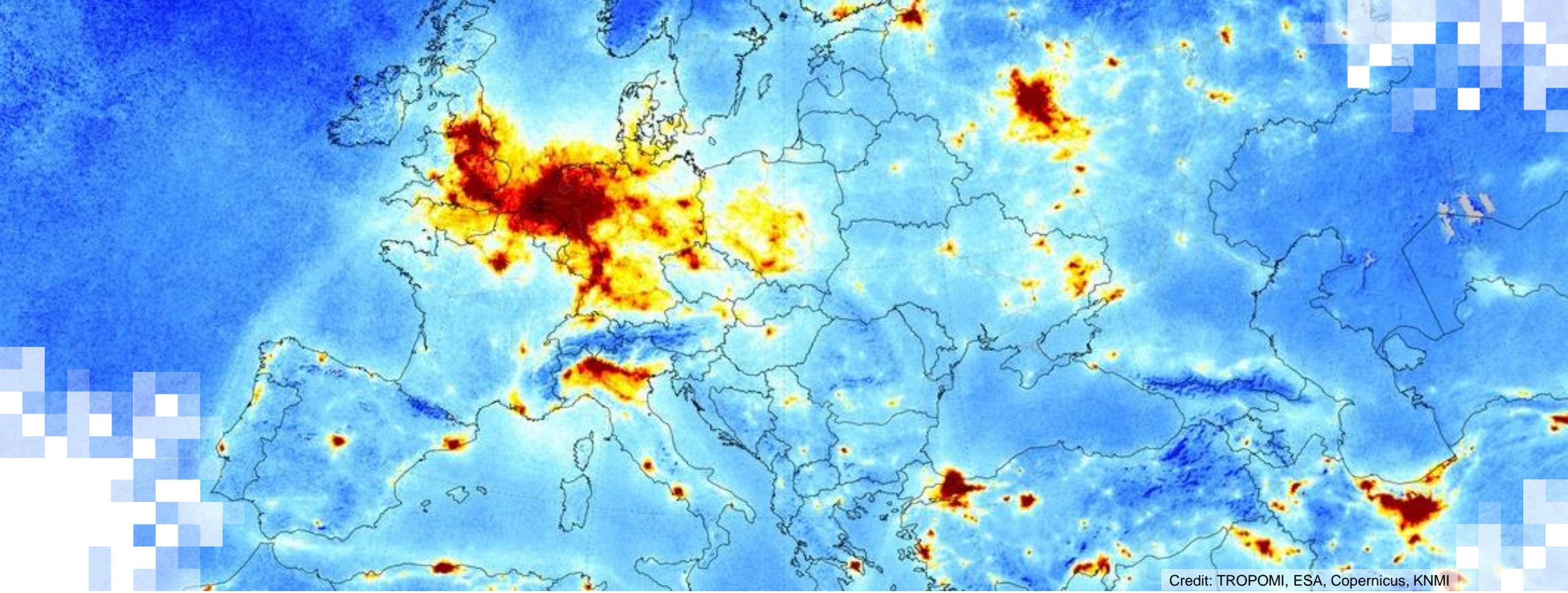


Fuente: Krotkov et al. (2017)

Columna vs. Superficie Relación y Tendencias



Fuente: Lamsal, L.N. et al. (2016)

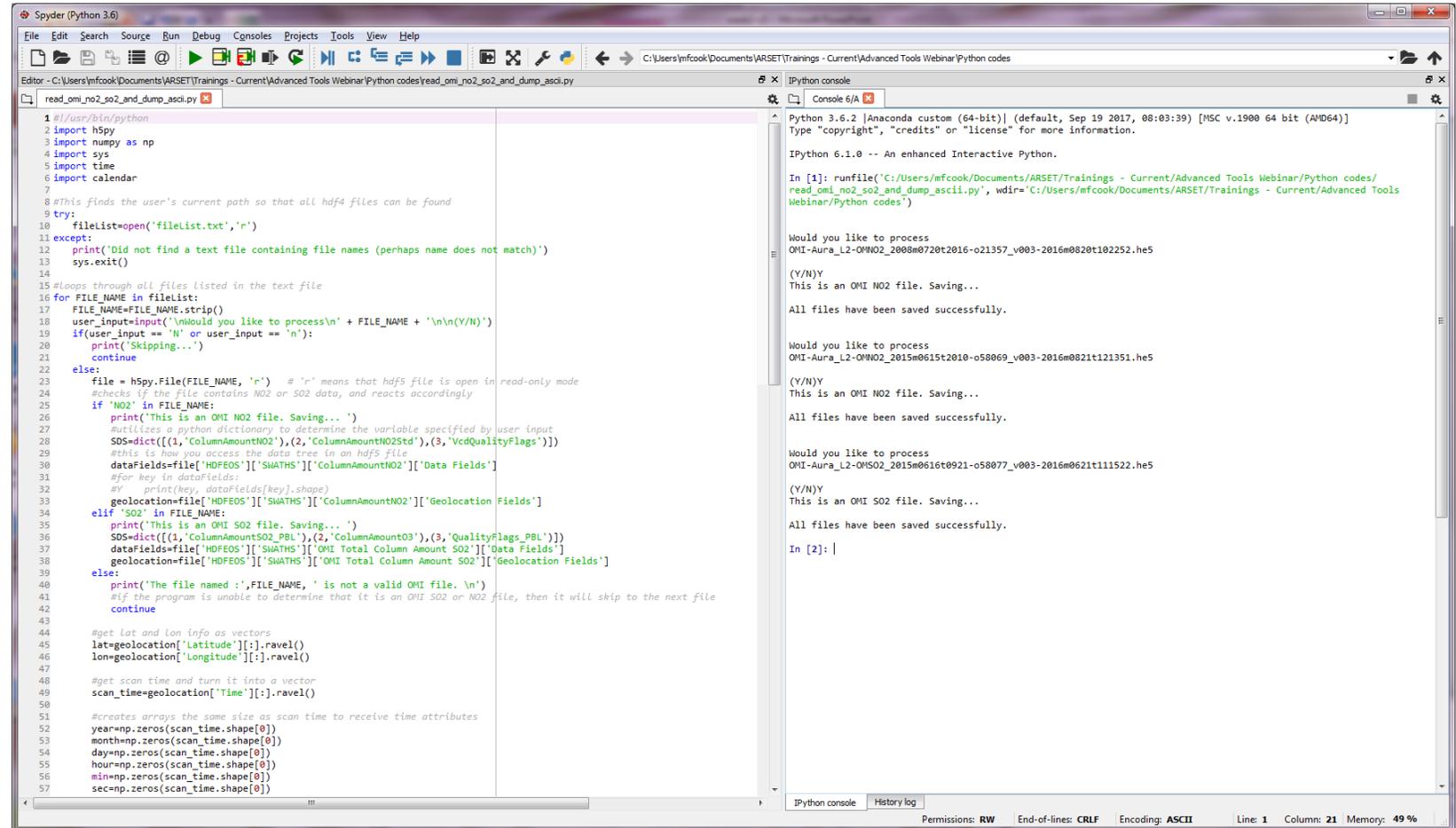


Exportar Variables HDF a CSV

Exportar Variables HDF OMI NO₂/SO₂ Como Salida a un Archivo CSV

read_omi_no2_so2_and_dump_ascii.py

- **Propósito:** leer un archivo de datos OMI nivel 2 de NO₂ o SO₂ en formato HDF y escribir ciertos SDS en un archivo csv (texto)



The screenshot displays the Spyder Python IDE interface. The left pane shows the source code for the script `read_omi_no2_so2_and_dump_ascii.py`. The code includes imports for `h5py`, `numpy`, `sys`, `time`, and `calendar`. It defines a function to read HDF files, check for NO₂ or SO₂ data, and export specific SDS to a CSV file. The right pane shows the IPython console output, which includes the Python version (3.6.2), the script path, and the execution results for three files: `OMI-Aura_L2-OMNO2_2008e0720t2016-o21357_v003-2016m0820t102252.he5`, `OMI-Aura_L2-OMNO2_2015m0615t2010-o50809_v003-2016m0821t121351.he5`, and `OMI-Aura_L2-OMSO2_2015m0616t0921-o58077_v003-2016m0621t111522.he5`. The console output shows that all files were processed successfully and saved.

```
1 #!/usr/bin/python
2 import h5py
3 import numpy as np
4 import sys
5 import time
6 import calendar
7
8 #This finds the user's current path so that all hdf4 files can be found
9 try:
10  fileList=open('fileList.txt','r')
11 except:
12  print('Did not find a text file containing file names (perhaps name does not match)')
13  sys.exit()
14
15 #Loops through all files listed in the text file
16 for FILE_NAME in fileList:
17  FILE_NAME=FILE_NAME.strip()
18  user_input=input('\nwould you like to process\n' + FILE_NAME + '\n\n(Y/N)')
19  if(user_input == 'N' or user_input == 'n'):
20  print('Skipping...')
21  continue
22  else:
23  file = h5py.File(FILE_NAME, 'r') # 'r' means that hdf5 file is open in read-only mode
24  #checks if the file contains NO2 or SO2 data, and reacts accordingly
25  if 'NO2' in FILE_NAME:
26  print('This is an OMI NO2 file. Saving... ')
27  #utilizes a python dictionary to determine the variable specified by user input
28  SDS=dict({'1','ColumnAmountNO2'},(2,'ColumnAmountNO2Std'),(3,'VcdQualityFlags'))
29  #this is how you access the data tree in an hdf5 file
30  dataFields=file['HDFEOS']['SWATHS']['ColumnAmountNO2']['Data Fields']
31  #for key in dataFields:
32  #    print(key, dataFields[key].shape)
33  geolocation=file['HDFEOS']['SWATHS']['ColumnAmountNO2']['Geolocation Fields']
34  elif 'SO2' in FILE_NAME:
35  print('This is an OMI SO2 file. Saving... ')
36  SDS=dict({'1','ColumnAmountSO2_PBL'},(2,'ColumnAmountO3'),(3,'QualityFlags_PBL'))
37  dataFields=file['HDFEOS']['SWATHS']['OMI Total Column Amount SO2']['Data Fields']
38  geolocation=file['HDFEOS']['SWATHS']['OMI Total Column Amount SO2']['Geolocation Fields']
39  else:
40  print('The file named ',FILE_NAME, ' is not a valid OMI file. \n')
41  #if the program is unable to determine that it is an OMI SO2 or NO2 file, then it will skip to the next file
42  continue
43
44 #get Lat and Lon info as vectors
45 lats=geolocation['Latitude'][:].ravel()
46 lons=geolocation['Longitude'][:].ravel()
47
48 #get scan time and turn it into a vector
49 scan_time=geolocation['Time'][:].ravel()
50
51 #creates arrays the same size as scan time to receive time attributes
52 year=np.zeros(scan_time.shape[0])
53 month=np.zeros(scan_time.shape[0])
54 day=np.zeros(scan_time.shape[0])
55 hour=np.zeros(scan_time.shape[0])
56 min=np.zeros(scan_time.shape[0])
57 sec=np.zeros(scan_time.shape[0])
```

Salida

```
OMI-Aura_L2-OMNO2_2008m0720t2016-o21357_v003-2016m0820t102252 - Notepad
File Edit Format View Help
Year,Month,Day,Hour,Minute,Second,Latitude,Longitude,ColumnAmountNO2,ColumnAmountNO2Std,vcdqualityFlags
2008.0.7.0.20.0.20.0.38.0.37.0,-74.78585052490234,-121.10774993896484,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-74.95336041250766,-116.02945709228516,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-75.00605010986378,-111.8778076171875,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-74.98909759521484,-108.39921569824219,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-74.93197631835938,-105.4294204711914,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-74.85054779052734,-102.85506439208984,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-74.75444030761719,-100.59456634521484,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-74.649658203125,-98.58740234375,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-74.54005432128906,-96.78759002685547,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-74.42811584472656,-95.15950775146484,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-74.3154525756836,-93.67511749267578,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-74.20308685302734,-92.31201171875,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-74.09165954589844,-91.05208587646484,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-73.98152923583984,-89.8805311103516,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-73.87289428710938,-88.78511810302734,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-73.76580810546875,-87.75564575195312,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-73.66024017333984,-86.78353118896484,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-73.55611419677734,-85.86150360107422,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-73.45328521728516,-84.98333740234375,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-73.35160827636719,-84.14366912841797,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-73.25088500976562,-83.33782958984375,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-73.15093994140625,-82.56175994873047,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-73.0671447539062,-81.8113103726146484,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-72.9525375366211,-81.08490753173828,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-72.8536605834961,-80.3780517578125,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-72.75472259521484,-79.6886978149414,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-72.65550231933594,-79.01445770263672,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-72.5557861328125,-78.35314178466797,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-72.45532989501953,-77.70271301269531,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-72.35392761230469,-77.06124877929688,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-72.25133514404297,-76.42691802978516,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-72.14730072021484,-75.79795837402344,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-72.04157257080078,-75.17264556884766,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-71.93387603759766,-74.54930114746094,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-71.82392883300781,-73.92621612548828,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-71.7114028930664,-73.30171203613281,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-71.59595489501953,-72.6740493774414,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-71.47721099853516,-72.04144287109375,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-71.354736328125,-71.40204620361328,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-71.22806549072266,-70.75390625,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-71.09664154052734,-70.094970703125,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-70.95985412597656,-69.42302703857422,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-70.81696319580078,-68.73568725385938,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-70.6671447539062,-68.0303726196289,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-70.50940704345703,-67.30422973632812,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-70.34258270263672,-66.55410766601562,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-70.16527557373047,-65.7764892578125,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-69.97581481933594,-64.96739959716797,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-69.77216339111328,-64.12232208251953,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-69.55181884765625,-63.23603820800781,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-69.31168365478516,-62.30249786376953,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-69.04785919189453,-61.314537048339844,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-68.75537109375,-60.26359176635742,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-68.4277572631836,-59.13920211791992,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-68.05646514892578,-57.928348541259766,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-67.62984466552734,-56.61440658569336,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-67.13152313232422,-55.175418853759766,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-66.53754425048828,-53.5811882019043,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-65.8108139038086,-51.78777313232422,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.37.0,-64.88919830322266,-49.7259521484375,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.39.0,-74.67247009277344,-120.99896240234375,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.39.0,-74.83917236328125,-115.95703887999453,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.39.0,-74.88825225830078,-111.83670806884766,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.39.0,-74.8129196166992,-105.4381738671875,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
2008.0.7.0.20.0.20.0.38.0.39.0,-74.73140716552734,-102.88363647460938,-1.2676506002282294e+30,-1.2676506002282294e+30,3.0
```

Este código guarda un archivo .csv, el cual se puede abrir con Excel, algún editor de texto, u otros códigos o software



Editando el Código

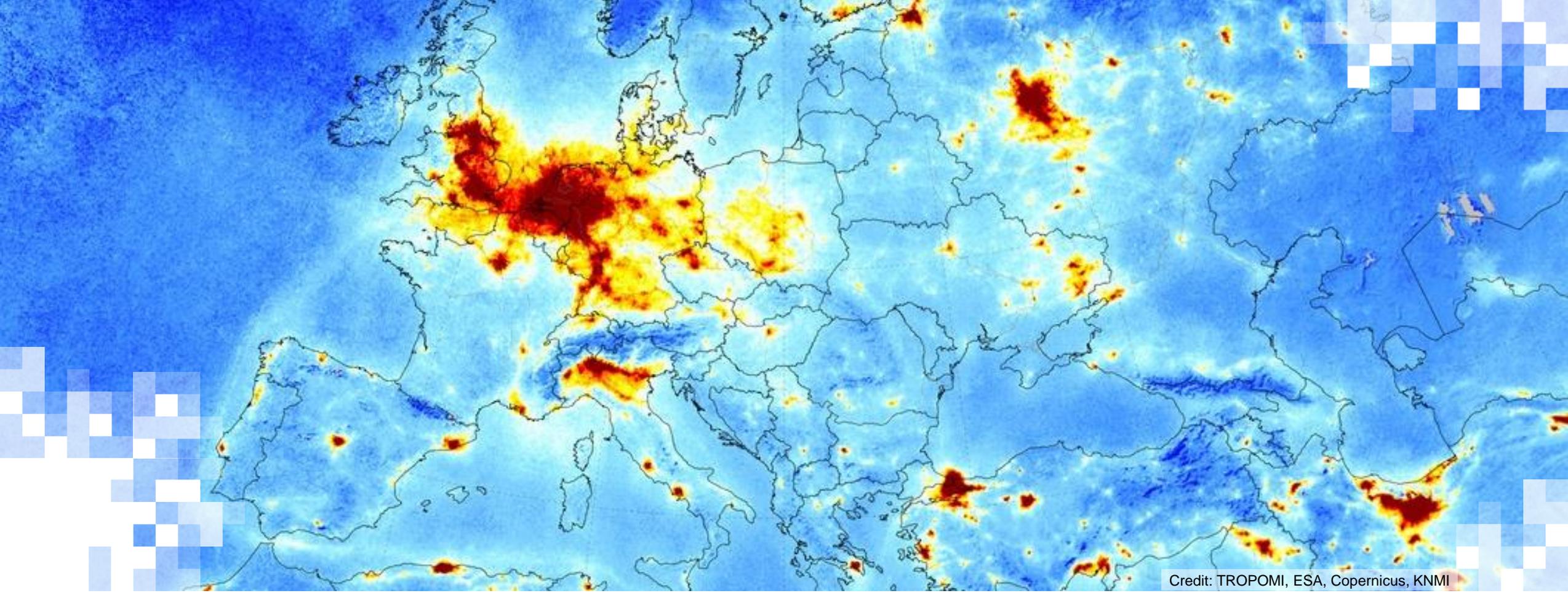
Cambie el SDS a ser escrito como salida

NOTE: Este código funcionará únicamente cuando todas las variables listadas sean la misma dimensión. Utilice el código “list SDS” para ver las dimensiones de las variables.

```
15 #Loops through all files listed in the text file
16 for FILE_NAME in fileList:
17     FILE_NAME=FILE_NAME.strip()
18     user_input=input('\nWould you like to process\n' + FILE_NAME + '\n\n(Y/N)')
19     if(user_input == 'N' or user_input == 'n'):
20         print('Skipping...')
21         continue
22     else:
23         file = h5py.File(FILE_NAME, 'r') # 'r' means that hdf5 file is open in read-only mode
24         #checks if the file contains NO2 or SO2 data, and reacts accordingly
25         if 'NO2' in FILE_NAME:
26             print('This is an OMI NO2 file. Saving... ')
27             #utilizes a python dictionary to determine the variable specified by user input
28             SDS=dict([(1, 'ColumnAmountNO2'),(2, 'ColumnAmountNO2Std'),(3, 'VcdQualityFlags')])
29             #this is how you access the data tree in an hdf5 file
30             dataFields=file['HDFEOS']['SWATHS']['ColumnAmountNO2']['Data Fields']
31             #for key in dataFields:
32             #Y     print(key, dataFields[key].shape)
33             geolocation=file['HDFEOS']['SWATHS']['ColumnAmountNO2']['Geolocation Fields']
34         elif 'SO2' in FILE_NAME:
35             print('This is an OMI SO2 file. Saving... ')
36             SDS=dict([(1, 'ColumnAmountSO2_PBL'),(2, 'ColumnAmountO3'),(3, 'QualityFlags_PBL')])
37             dataFields=file['HDFEOS']['SWATHS']['OMI Total Column Amount SO2']['Data Fields']
38             geolocation=file['HDFEOS']['SWATHS']['OMI Total Column Amount SO2']['Geolocation Fields']
39         else:
40             print('The file named :,FILE_NAME, ' is not a valid OMI file. \n')
41             #if the program is unable to determine that it is an OMI SO2 or NO2 file, then it will skip to the next file
42             continue
43
```

Aplicaciones

- Este es un ejemplo de código para leer y extraer datos OMI Nivel 2 de NO₂ y SO₂
- Se puede modificar el código para extraer varios SDS en un solo archivo .csv
- El código se puede modificar fácilmente para extraer datos sobre alguna región
- El archivo de salida se puede abrir en Excel o cualquier otra herramienta de análisis de datos



Preguntas y Respuestas